# Towards Collaborative Anti-Money Laundering Among Financial Institutions

Anonymous Author(s)*

## ABSTRACT

Money laundering is the process that intends to legalize the income derived from illicit activities, thus facilitating their entry into the monetary flow of the economy without jeopardizing their source. It is crucial to identify such activities accurately and reliably in order to enforce anti-money laundering (AML).

Despite considerable efforts to AML, a large number of such activities still go undetected. Rule-based methods were first widely used in the early days and still be widely used in existing detection systems. With the rise of machine learning, graph-based learning methods have gained prominence in detecting illicit accounts by analyzing money transfer graphs between accounts. However, existing approaches work based on the prerequisite that the transaction graph is centralized, while in practice, money laundering activities usually span multiple financial institutions. Due to regulatory, legal, commercial, and customer privacy concerns, institutions tend not to share data, limiting their utility in practical usage. In this paper, we propose the *first* algorithm that supports performing AML over multiple institutions while protecting the security and privacy of local data.

To evaluate, we construct Alipay-ECB, a real-world dataset comprising digital transactions from Alipay, the world's largest mobile payment platform, alongside transactions from E-Commerce Bank (ECB). The dataset includes over 200 million accounts and 300 million transactions, covering both intra-institution transactions and those between Alipay and ECB. This makes it the largest real-world transaction graph available for analysis. The experimental results demonstrate that our methods can effectively identify cross-institution money laundering subgroups. Additionally, experiments on synthetic datasets also demonstrate that our method is efficient, requiring only a few minutes on datasets with millions of transactions.

## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

## KEYWORDS

anti-money laundering, collaborative learning



*Conference acronym 'XX, June 03–05, 2018, Woodstock, NY*



ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

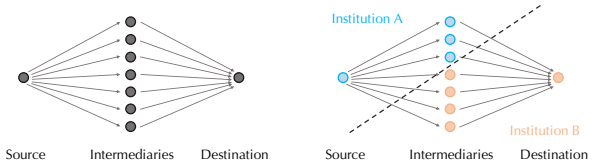



**Figure 1: (a) Scatter-gather pattern money laundering; (b) Scatter-gather distributed across two institutions.**

**ACM Reference Format:**
Anonymous Author(s). 2018. Towards Collaborative Anti-Money Laundering Among Financial Institutions. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Money laundering is a process that attempts to conceal or disguise the origins of dirty money derived from illicit activities, making it appear as if the funds have been obtained through legitimate means [15]. It typically consists of three primary steps: a *placement* step first introduces the dirty money into existing financial systems; a *layering* step then carries out complex transactions to hide the source of the funds; and a *integration* step withdraws the fund from a destination bank account before using it for legitimate activities [14]. The transaction relationship of accounts can be represented as a graph, where an individual account is denoted as a node, and transactions between two accounts are denoted as edges. Due to the distinctive nature of money laundering activities, the transaction graph associated with money launderers exhibits a unique pattern known as **scatter-gather** [3, 5, 13], as illustrated in Fig. 1a.

It is the responsibility of financial institutions to conduct *anti-money laundering* (AML): diligently monitor transactions, take necessary actions like shutting down or imposing restrictions on suspicious accounts, and promptly report any suspicious activities through to law enforcement agencies. To detect money laundering activities, a common idea is to identify the ultimate beneficiary, which refers to the individual or entity that ultimately receives the funds, even if those funds have been obscured through multiple layers of transactions [15]. To achieve that, a simple approach is to calculate the ratio to which funds in one account originate from another account [13]. If the ratio exceeds a predefined threshold, it indicates a potential association between the two accounts, raising suspicions of money laundering activities with one account being the source and the other the destination.

However, money laundering has evolved into a highly sophisticated process, spanning across multiple financial institutions s.t. the subgraph within one institution appears to be normal (Fig. 1b).

As a result, relying solely on the transaction graph within a single institution for AML is no longer sufficient. A straightforward solution is to combine the transaction graphs from multiple institutions. However, due to regulatory, legal, commercial, and customer privacy concerns, institutions tend not to share data.

**Our contribution.** In this paper, we make the *first* step towards collaborative AML, which allows multiple institutions to jointly conduct AML without exposing their individual transaction graphs.

Our primary contribution lies in the introduction of a novel algorithm for scatter-gather subgraph mining, specifically tailored to suit the collaborative setting. In more detail, this algorithm first employs a breadth-first search (BFS) approach for each node to identify a set of cross-institution transactions associated with that node, which can be either scattered from or gathered towards the node. If two nodes, belonging to different institutions, share the same set of cross-institution transactions, it indicates a potential scatter-gather relationship within a money laundering subgraph, with one node being the source and the other being the destination. Building upon this observation, the algorithm considers two institutions, denoted by $\mathcal{P}_A$ and $\mathcal{P}_B$, and iterates through their respective nodes ($\{N_1^A, N_2^A, \ldots, N_n^A\}$ and $\{N_1^B, N_2^B, \ldots, N_n^B\}$) to identify the sets of cross-institution transactions: $\mathcal{S}^A = \{S_1^A, S_2^A, \ldots, S_n^A\}$ and $\mathcal{S}^B = \{S_1^B, S_2^B, \ldots, S_n^B\}$, where e.g., $S_i^A$ is the set of cross-institution transactions associated with node $N_i^A$. If two sets $S_i^A$ and $S_j^B$ exhibit a high degree of similarity, it suggests that $N_i^A$ and $N_j^B$ are potentially involved in scatter-gather activities within a money laundering subgraph.

This approach requires $\mathcal{P}_A$ and $\mathcal{P}_B$ to exchange $\mathcal{S}^A$ and $\mathcal{S}^B$, and measure the similarity between each pair (e.g., $S_i^A$ and $S_j^B$). This is costly in terms of both communication and computation. To solve the problem, we use locality-sensitive hashing (LSH) [30] and Bloom filter [27] to minimize the amount of information to be exchanged between $\mathcal{P}_A$ and $\mathcal{P}_B$. LSH enables the estimation of similarity between two sets by comparing the minimum hash values of their elements. Combined with Bloom filters, the approach transforms pairwise comparisons into a process of testing the presence of an element within a Bloom filter. The Bloom filter is memory-efficient, and this testing process is computationally efficient.

Specifically, an LSH is computed for each set, resulting in $\{lsh_1^A, lsh_2^A, \ldots, lsh_n^A\}$ and $\{lsh_1^B, lsh_2^B, \ldots, lsh_n^B\}$. Notice that $lsh_i^A = lsh_j^B$ if $S_i^A$ and $S_j^B$ exhibit a high degree of similarity. Next, one institution, say $\mathcal{P}_A$, inserts $\{lsh_1^A, lsh_2^A, \ldots, lsh_n^A\}$ into a bloom filter $BF_A$, and transfers $BF_A$ to $\mathcal{P}_B$; $\mathcal{P}_B$ iterates through $\{lsh_1^B, lsh_2^B, \ldots, lsh_n^B\}$ to check if each $lsh^B$ is present in $BF_A$. If $lsh_j^B$ is found in $BF_A$, $\mathcal{P}_B$ learns that $N_j$ is one end node in the scatter-gather activity. At this stage, $\mathcal{P}_B$ reveals the corresponding $lsh_j^B$ to $\mathcal{P}_A$, enabling $\mathcal{P}_A$ to identify the other end node in the scatter-gather activity. By leveraging this optimization, the communication overhead is significantly reduced as it only requires the transfer of a bloom filter. Moreover, by comparing against a bloom filter, the computational complexity is reduced to $O(n)$, rather than $O(n^2)$ when comparing each pair individually.

To evaluate whether our methods can detect money laundering activities across multiple institutions in a real-world setting, we construct Alipay-ECB, a multi-institution transaction dataset that includes digital currency transactions from Alipay and E-Commerce Bank (ECB) users. The dataset contains over 200 million accounts and 300 million transactions. To the best of our knowledge, it is the largest real-world transaction dataset available.

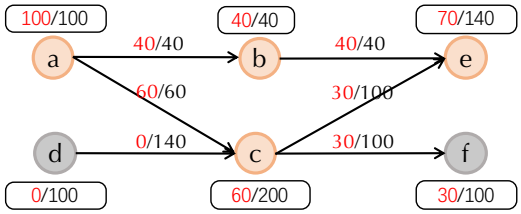



**Figure 2: Illustration of centralized scatter-gather mining. Consider $a$ as the source and designate all the funds flowing out from $a$ as illicit money, visually represented in red. The enclosed numbers within boxes indicate the money possessed by the nodes, while numbers above the line represent the money involved in a transaction. The orange nodes represent the detected money laundering nodes, which have an illicit funds ratio greater than 0.3.**

By analyzing the dataset, we find that money laundering groups possess a much more intricate structure in real-world settings, encompassing multiple simple patterns such as fan-in, fan-out, cycles, random, and bipartite, etc. However, our method can effectively identify money laundering subgroups. Experiments on synthetic datasets also demonstrate our methods can effectively and efficiently identify money laundering subgroups.

## 2 PRELIMINARIES

This section provides the necessary background and preliminaries for understanding this paper. The frequently used notations are presented in Table 3.

### 2.1 Scatter-Gather Mining

In order to detect money laundering transaction subgraphs of scatter-gather patterns, a simple approach is to examine cases where a significant amount of money flows out of one account and gets aggregated in other accounts [13]. We refer to this method as *Centralized Scatter-Gather Mining*. To illustrate, let's consider an example where there's a node $j$ that receives 80% of the money flowing out from node $i$. In this case, it's possible that both nodes $i$ and $j$, along with the nodes in-between, are involved in a potential money laundering activity, with $i$ acting as the source within the subgraph, and $j$ serving as the destination.

To determine how much of the money received by node $j$ comes from node $i$, the method utilizes a tracking mechanism based on the transaction graph. This involves marking the outflow money from node $i$ as suspected money and tracing their movement within the graph. When node $i$ sends money to another node $v$, the marked money is transferred to $v$. Similarly, if node $v$ subsequently sends money to node $j$, the marked money is also transferred to node $j$. In the context of the method, two principles govern the flow of marked money in downstream nodes, considering that money is divisible. Denote $M_{in}^j, M_{out}^j$ as total inflow and outflow of node $j$

separately, and $m^j_{in}, m^j_{out}$ as marked inflow and outflow included in $M^j_{in}, M^j_{out}$ that satisfy $m^j_{in} \le M^j_{in},\ m^j_{out} \le M^j_{out}$.

We have the following principles to calculate $m^j_{in}$:

(1) For a node whose inflow money involves marked money, if $M^j_{in} > M^j_{out}$, then $m^j_{out} = m^j_{in} \frac{M^j_{out}}{M^j_{in}}$; if $M^j_{in} \le M^j_{out}$, then $m^j_{out} = m^j_{in}$.
(2) The marked inflow money of a node is the sum of marked money received from other nodes.

After getting the value of $m^j_{in}$, we calculate the ratio of inflow money from $i$ to $j$ as $r_{ij} = \frac{m^j_{in}}{m^i_{out}}$.

Figure 2 illustrates an example of applying the method by considering node $a$ as the source node and discovering the scatter-gather pattern it is involved in. By setting the threshold to 40%, we identify three suspected money laundering nodes $b$, $c$, and $e$, which contain 40%, 60% and 70% of marked money, respectively.

## 2.2 MinHash

MinHash [29] is a technique to estimate how similar two sets are, where the similarity is defined in terms of the Jaccard similarity coefficient. Specifically, let $A$ and $B$ are two sets. The Jaccard index is defined to be the ratio of the number of elements of their intersection and the number of elements of their union:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{1}$$

Let $H$ denote the minhash function that maps a set to a real number; it has the property

$$Pr[H(A) = H(B)] = J(A,B). \tag{2}$$

That is, the probability that $H(A) = H(B)$ is true is equal to the similarity $J(A,B)$.

The details of the MinHash algorithm is following: Given a hash function $h$ that maps the members of a set $U$ to real numbers, and *perm* which is a random permutation of the elements of $U$. For any set $S \subset U$, $H$ is defined as the minimum value of $h(perm(x))$, i.e.,

$$H(S) := min\ h(perm(x)). \tag{3}$$

Let $r$ be a random variable that is 1 when $H(A) = H(B)$ and 0 otherwise, $r$ is the unbiased estimator of $J(A,B)$, i.e., $E(r) = J(A,B)$. The MinHash scheme reduces this variance by averaging together several variables constructed in the same way, such as by applying multiple hash functions. To estimate $J(A,B)$, let $n$ be the number of hash functions for which $H(A) = H(B)$, $\frac{n}{K}$ is the estimate, where $K$ is the total number of hash functions used. This estimate is the average of $K$ random variables $r$s, each of which is the unbiased estimator of $J(A,B)$. Hence, the average is also unbiased. By standard deviation for sums of the variables, the similarity estimation error is $O(1/\sqrt{K})$.

## 2.3 Bloom filter

A Bloom filter [27] is a memory-efficient data structure that is used to test whether an element is present in a set. The price paid for the efficiency is that Bloom filter is a probabilistic data structure: It tells us that the element either *definitely* is not in the set or *may be* in the set. In other words, false positive matches are possible, but false negatives are not.

A Bloom filter is an array of $m$ bits with all positions set to 0 when it is empty. There are also $k$ hash functions, each of which maps or hashes each element in a set to one of the $m$ positions uniformly. To *add* an element, we simply feed it to each of the $k$ hash functions to get $k$ array positions and set the bits at all these positions to 1. To *query* an element (test whether it is in the set), hash it using the identical $k$ hash functions to get $k$ array positions. If any of the $k$ positions are 0, the element is *definitely* not in the set. If all are 1, then the element is either in the set or the bits were set to 1 when inserting other elements by chance, resulting in a false positive. The false positive error $\epsilon$, the size of Bloom filter $m$, and the number of hash functions $k$ are related in the following way:

$$k = -\log_2 \epsilon,\ m = -\frac{n \ln \epsilon}{(\ln 2)^2}, \text{and } \epsilon = (1 - e^{-\frac{kn}{m}})^k \tag{4}$$

With $m$ increases, the false positive probability $\epsilon$ decreases.

# 3 PROBLEM STATEMENT

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ be a money transaction graph, where $\mathcal{V}$ is the vertex set represents accounts, $\mathcal{E}$ is the edge set represents transactions, and $\mathcal{X} \in \mathbb{R}^d$ is the feature matrix of all edges. An edge $(i,j) \in \mathcal{E}$ indicates that the account $i$ transfers money to $j$ and the corresponding $\vec{x} \in \mathcal{X}$ indicates the attributes of the transaction, such as the amount of money, the time, to name a few. In this paper, we mainly focus on two attributes: the amount of money and whether the transaction is an external transaction, denoted as $a$ and $c$ separately. Specifically, $\vec{x} = [a, c]^\top$. For ease of presentation, we denote $\vec{x}_{i \to j}$ the attributes for the transaction from $i$ to $j$.

In our setting of collaborative learning, we consider two institutions $\mathcal{P}_A$ and $\mathcal{P}_B$; each holds a subgraph $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A, \mathcal{X}_A)$ and $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B, \mathcal{X}_B)$, where $\mathcal{V}_i, \mathcal{E}_i, \mathcal{X}_i$ are subsets of $\mathcal{V}, \mathcal{E}, \mathcal{X}$, separately. In the rest of the paper, we use the notations $p$ and $q$ to denote the indices of the two institutions. Specifically, $\mathcal{P}_p$ refers to one institution and $\mathcal{P}_q$ to the other.

To comply with Know Your Customer (KYC) standards [28], financial institutions are required to gather basic information about both the initiator and recipient of each transaction. This rule remains applicable even when accounts are held across different institutions. Based on this requirement, we assume an overlap between $\mathcal{V}_A$ and $\mathcal{V}_B$. The overlapping nodes represent accounts involved in cross-institution transactions between $\mathcal{P}_A$ and $\mathcal{P}_B$.

We further assume that the overlapping accounts are recorded with identical identifiers by both institutions. This identification can be performed privately through multi-party private set intersection methods [9], which is orthogonal to our paper.

Given the above setting, we aim to discover money laundering groups of typologies presented in figure 1a based on two subgraphs $\mathcal{V}_A$ and $\mathcal{V}_B$.

# 4 METHODS

In this section, we present in detail how our collaborative AML algorithm, named collaborative scatter-gather mining (CSGM), is designed. We begin by transforming the centralized scatter-gather mining method into the one that can be applied to two subgraphs

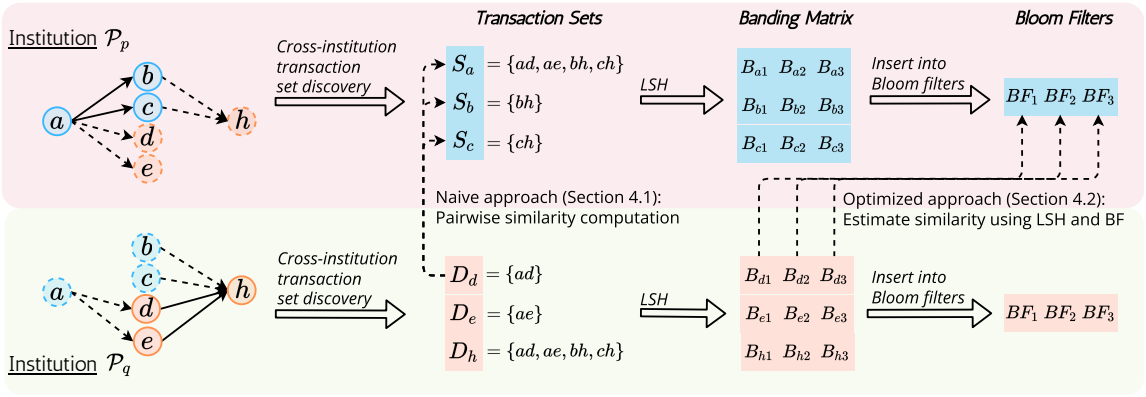



**Figure 3: Workflow of CSGM. The dotted lines on the graphs indicate cross-institution transactions.**

as defined in Section 3 owned by different institutions. The method enables the detection of money laundering nodes distributed across multiple institutions, particularly when the source and destination nodes belong to different institutions.

We further enhance the method by making use of Locality-sensitive hashing (LSH) [30] and Bloom filter [27] to minimize communication costs and improve efficiency. Figure 3 presents the workflow of CSGM.

## 4.1 Collaborative Scatter-Gather Mining

In the scatter-gather pattern of money laundering, money is transferred from a source to a destination through multiple transactions involving many adversarial middle nodes. When the source and the destination are located in different institutions, it implies that money laundering activities transfer money to another institution via cross-institution transactions, as shown in Figure 1b.

The key idea behind our method is that the set of cross-institution transactions scattered from the source is identical to the set of cross-institution transactions gathered at the destination when the source and destination are involved in the same money laundering subgraph. Therefore, by comparing the sets of transactions identified by both institutions, we can effectively detect money laundering subgraphs in which both source and destination are implicated.

Specifically, let $\mathcal{S}_p \leftarrow [S_i \mid i \in \mathcal{V}_p]$ and $\mathcal{D}_p \leftarrow [D_i \mid i \in \mathcal{V}_p]$ for $p \in \{A, B\}$ denote the sets of all cross-institution transactions associated with $\mathcal{P}_p$, where $S_i$ and $D_i$ represent the sets obtained through scattering from or gathering to node $i$, respectively. $\mathcal{P}_p$ transmits both $\mathcal{S}_p$ and $\mathcal{D}_p$ to institution $\mathcal{P}_q$, ensuring that both $\mathcal{P}_A$ and $\mathcal{P}_B$ possess all relevant sets. By independently comparing the similarity between any two sets $S_i \in \mathcal{S}_p$ and $D_j \in \mathcal{D}_q$, each institution can identify sources or destinations involved in money laundering activities. Specifically, $\mathcal{P}_p$ can detect sources by comparing sets from $\mathcal{S}_p$ with those from $\mathcal{D}_q$, and similarly, identify destinations by comparing sets from $\mathcal{D}_p$ with those from $\mathcal{S}_q$. Note that we filter out the discovered sets of small size (setting the threshold to 4-7 in our experiments), considering that money laundering groups are typically huge to conceal substantial amounts of money. Once all suspicious sources and destinations are identified, intermediate nodes can be readily located by tracing the transactions that are scattered from sources or gathered to destinations within the local subgraph.

**Cross-institution Transaction Set Discovery.** To find the set of cross-institution transactions, each institution employs the BFS approach for each node to find transactions scattered or gathered from the node and determine if they are cross-institution transactions. Specifically, it starts from a specific node and loops all neighbor nodes to identify cross-institution transactions originating from the node until either all relevant transactions are found or the maximum depth is reached. Let $\mathcal{F}$ represent the algorithm, and we denote the discovery process as $S_i \leftarrow \mathcal{F}(i, \mathcal{G}, T)$. Here, $S_i$ is the set of cross-institution transactions scattered from node $i$, $\mathcal{G}$ is the local transaction graph, and $T$ denotes the maximum depth allowed. When aiming to discover the gathered transaction sets, we can simply transform $\mathcal{G}$ into a new graph $\mathcal{G}'$ with the inverse direction. Specifically, $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \mathcal{X})$, where $\mathcal{E}' = \{(j, i) | (i, j) \in \mathcal{E}\}$. By performing the same algorithm on $\mathcal{G}'$, we construct the reversed transaction set as $D_i \leftarrow \mathcal{F}(i, \mathcal{G}', T)$. Algorithm 3 in Appendix presents the procedure.

## 4.2 Optimization for Distributed Scatter-Gather Mining

Applying the distributed scatter-gather mining algorithm directly is both communication- and computation-intensive, as it requires institutions to exchange multiple transaction sets ($O(n)$, where $n$ is the number of nodes) and perform pairwise comparisons among them, which is $O(n^2)$. To address the challenge, we propose an optimized algorithm using LSH [30] and Bloom filters [27]. LSH enables the estimation of similarity between two sets by comparing the minimum hash values of their elements, and by inserting the results of all sets (either from $\mathcal{S}$ or $\mathcal{D}$) into a bloom filter, we transform pairwise comparisons into a more efficient process of testing whether an element exists within the Bloom filter.

Specifically, institution $\mathcal{P}_p$ first performs LSH on all sets. The results are then inserted into $K$ Bloom filters, where $K$ is determined by the length of the LSH value. The Bloom filters are then shared with another institution, $\mathcal{P}_q$. By querying the Bloom filter with the LSH of $\mathcal{P}_q$'s local set, which is likely to match those of other sets with high similarity, $\mathcal{P}_q$ can efficiently detect the existence of a

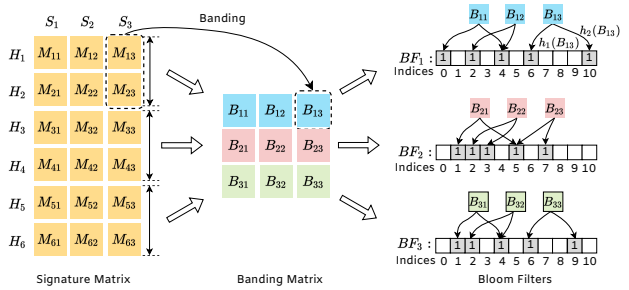



**Figure 4: Example of inserting transaction sets into Bloom filters. We consider three sets $S_1$, $S_2$, and $S_3$ and 6 MinHash functions. The band width $r = 2$**

similar set, thereby determining whether the corresponding node is involved in potential money laundering activities. As it requires only the transfer of Bloom filters, the optimization significantly reduces communication overhead, Moreover, the computational complexity is reduced to $O(Kn), k << n$, as opposed to $O(n^2)$ when performing pairwise comparisons.

Next, we provide a detailed explanation of how sets are inserted into Bloom filters. We then introduce two methods, namely Probability-Based Similar Set Detection and Similarity-Based Similar Set Detection, to detect similar sets using Bloom filters.

*4.2.1 Inserting sets into Bloom filters.* We adopt the MinHash algorithm (cf. Section 2.2) as the approach to implement LSH. Take $\mathcal{S}^p$ as an example, $\mathcal{P}_p$ first employs $m$ distinct minhash functions $H$ (cf. Equation 3) on each set $S_i \in \mathcal{S}_p$ resulting in a signature matrix $M_S^p$ with $m$ rows and $|\mathcal{S}^p|$ columns, where $|\cdot|$ denotes the number of sets in $\mathcal{S}_p$. Each row of the matrix represents applying the same minhash function to all sets in $\mathcal{S}_p$, and each column represents applying all minhash functions to the same set.

A banding technique then be applied to the matrix. Specifically, we divide the matrix into bands, each containing $r$ rows of the matrix, resulting in a total of $K = m/r$ bands. Each column of a band, which is composed of the result of applying $r$ minhash functions to one set, can be treated as a result of applying LSH on the set. If two sets have the Jaccard similarity of $s$, then the probability that their columns within the same band are equal is $s^r$. By mapping each column to a distinctive signature, for example, by utilizing the MD5 function [31], each band can be treated as one row of the banding matrix. We denote it as $B_S^p$. We then insert each band into a Bloom filter (cf. Section 2.3), resulting in $K$ Bloom filters $BF_S^p[1], ..., BF_S^p[K]$. When the context is clear, we omit the superscripts and subscripts, and represent each Bloom filter as $BF_k, k \in \{1, ..., K\}$.

We note that to guarantee the LSH of two similar sets are equal with high probability, $\mathcal{P}_p$ and $\mathcal{P}_q$ are required to use the same MinHash functions on $\mathcal{S}_p$ and $\mathcal{D}_q$. Figure 4 presents an example of inserting three sets $S_1$, $S_2$, and $S_3$ into three Bloom filters, with the band with $r = 2$.

*4.2.2 Probability-based similar set detection.* With the received Bloom filters, institutions can detect whether the node is involved

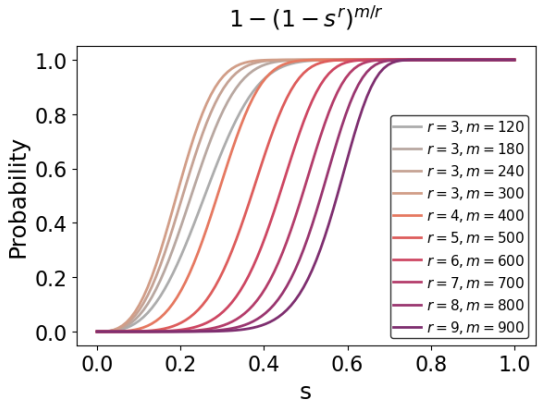



**Figure 5: The probability calculated with Equation 5 with different $r$ and $m$.**

in money laundering activities by querying the existence of the band values in corresponding Bloom filters. Specifically, for a set $S_i \in \mathcal{S}^p$, denote its band values as $B_i$, which are a column in the banding matrix. $\mathcal{P}_p$ query the existence of each $B_{ki}$ in the corresponding Bloom filter $BF_k$. Theoretically, if there exists a set $D_j \in \mathcal{D}^q$ that exhibits a similarity of $s$ with $S_i$, the probability that at least one Bloom filter contains $B_{ki}$ is:

$$1 - (1 - s^r)^K, \tag{5}$$

where $K = m/r$. As shown in Figure 5, by appropriately selecting values for $m$ and $r$, this probability can be adjusted to be close to 1 or 0, depending on the level of similarity. For example, when the threshold is 0.4, we set $r = 4$ and $m = 400$ so the probability is about 0.92. Consequently, if at least one $B_{ki}$ tested exists in $BF_k$, we treat the corresponding node of $S_i$ as a potential source within a money laundering subgroup.

With the probability-based similar set detection, we denote our AML method as **Prob-CSGM** and Algorithm 1 presents the pseudo-code of the method.

**Algorithm 1:** IsSimilarSetProb

**Input:** $S_i$, $BF_k, k \in \{1, ..., K\}$
**Output:** $h$, $h = 1$ indicates the set is a similar set

```
h = 0
for k ∈ {1, ..., K} do
    B_ki ← BANDING({H_t(S) | t ∈ [(k − 1)r, kr]})
    if B_ki ∈ BF_k) then
        h = 1
        break
    end
end
return h
```

*4.2.3 Similarity-based similar set detection.* While the probability-based method enables the detection of source/destination nodes involved in money laundering activities, it suffers from a high false positive rate, as dissimilar sets may still be detected in at least one

Bloom filter. Conversely, even when the threshold is increased, there remains a possibility that similar sets may go undetected. To address this limitation, we propose an alternative approach to estimate the similarity directly.

Recall that each institution divides the signature metrics $M$ into $K$ bands. The probability that any two columns within a band are identical is given by $s^r$, where $s$ is the similarity we aim to estimate. A straightforward approach is setting $r = 1$ and estimating the similarity by calculating the ratio of Bloom filters that contain the band value of the transaction set to the total number of Bloom filters. Formally, let $t_i$ be a random variable that equals 1 if $B_{ki}$ is present in the $k$-th Bloom filter $BF_k$ and 0 otherwise. The estimated Jaccard similarity is then given by $\frac{1}{m}\sum_{k=1}^{K} t_i$. A set $S_i$ is flagged as a money laundering set if this estimated similarity exceeds a predefined threshold $\tau$.

However, applying the above method introduces significant bias in the similarity estimation. This bias occurs because a Bloom filter stores hash values from all sets $D \in \mathcal{D}$, and multiple sets may share overlapping elements with $S_i$. As a result, this overlap leads to an overestimation of similarity, as the Bloom filter cannot differentiate between the contributions of different sets that share elements with $S_i$. Specifically, assume that sets $D_1, \ldots, D_Q \in \mathcal{D}$ have overlapping elements with $S_i$, such that $J(S_i, D_q) > 0$ for $q \in \{1, ..., Q\}$. Let $A_q$ denote the event that $B_i^S = B_q^D$. The probability of this event occurring is given by $\Pr(A_q) = \frac{|S_i \cap D_q|}{|S_i \cup D_q|}$, which is the Jaccard similarity between $S_i$ and $D_q$. Let $z$ be a random variable that equals 1 if $B_i^S$ is detected in the Bloom filter and 0 otherwise, we have

$$\begin{aligned} Pr[z=1] &= Pr(\bigcup_{k=1}^{K} A_k) \\ &= S_1 - S_2 + \cdots + (-1)^{n-1} S_n \\ &\leqslant \min\{S_1, 1\} \end{aligned} \tag{6}$$

where $S_k = \sum_{1 \leqslant i_1 < \cdots < i_k \leqslant n} P\left(A_{i_1} \cap \cdots \cap A_{i_k}\right)$.

To solve the problem, we propose estimating $s^r, r \geq 1$ instead of $s$. The reason is that, to detect money laundering groups, we can focus on the largest similarity between $S_i$ and any set $D_q$, defined as $P := \max\{Pr(A_1), \ldots, Pr(A_q)\}$. By estimating $s^r$, the probability that less similar sets produce equivalent banding values to $S_i$ is reduced, resulting in a more accurate estimation of $P^r$, and hence $P$. For example, consider two sets, $D_1$ and $D_2$, with similarities of 0.8 and 0.2 with $S_i$, respectively. Directly estimating $s$ can introduce a bias of up to 0.2, as $Pr[z = 1] - 0.8 < 0.2$. When $r = 2$, the probability that the banding values are equal is 0.81 for $D_1$ and 0.04 for $D_2$. This results in a more accurate estimation of $P$, as $\sqrt{Pr[z=1]} - 0.8 < \sqrt{0.64 + 0.04} - 0.8 \simeq 0.02$. A formal analysis is presented as follow:

Theorem 1. *Suppose that $X_1, ..., X_N$ are a sequence of real values with $0 \leq X_N \leq ... \leq X_1 \leq 1$. Then $\forall \varepsilon > 0$, when $r > \log_p(\frac{\varepsilon}{X_1(N-1)})$,*

$$(\sum_{i=1}^{N} X_1^r)^{1/r} - X_1 \leq \varepsilon.$$

The proof is presented in Appendix A.1. Based on the analysis, we estimate $P^r$ as $l/K$, where $l = \sum_{k=1}^{M/r} \mathbb{1}[B_{ki} \in BF_k]$, which represents the number of occurrences where $B_{ki}$ is found in $BF_k$. We identify sets involved in money laundering if their similarities exceed a predefined threshold. We can define this threshold as $\tau^r$ or estimate the similarity as $(l/K)^{1/r}$.

We denote the method as **Sim-CSGM** and present in the pseudocode in Algorithm 2.

**Algorithm 2:** IsSimilarSetSim

**Input:** Query set $S_i$, Bloom filters $BF_k, k \in \{1, .., K\}$
**Output:** $h$, $h = 1$ indicates the set is a similar set

```
for k ∈ {1, ..., M/r} do
    B_ki ← BANDING({H_t(S) | t ∈ [(k − 1)r, kr]})
    if B_ki ∈ BF_k then
        t_k = 1
    else
        t_k = 0
    end
end
s_i = (Σ_{k=1}^{K} t_k r)/m
if s_i ≥ τ^r then
    h=1
else
    h=0
end
return h
```

## 5 EXPERIMENTS

In this section, we conduct extensive experiments to verify the effectiveness and efficiency of CSGM.

### 5.1 Experimental setting

**Dataset.** To evaluate whether our methods can detect money laundering activities in practice, we construct a real-world dataset called *Alipay-ECB*, based on daily transactions recorded on Alipay [1] and E-Commerce Bank [2]. It comprises over 200 million accounts and 300 million transactions. To the best of our knowledge, it is the largest transaction dataset that tracks currency flow in the real world, providing a comprehensive reflection of money laundering activities. Detailed information on the dataset is provided in Appendix A.2. We also conduct experiments on synthetic datasets that simulates transactions for money laundering activities. We utilize AMLSim [20] and AMLWorld [3], which supports building a multi-agent simulator of anti-money laundering and has been widely used in previous works [5, 23, 26**?** ]. For AMLSim [20], we synthetic two datasets with 1 000 000 nodes, and simulate two scenarios where two institutions have balanced transaction subgraphs or unbalanced subgraphs. We refer to *AMLSim_bal* and *AMLSim_unb*, respectively. For AMLWorld [3], we choose two datasets of size 5 million and 7 million, respectively, for experiments. We refer to *AMLWorld_HI* and *AMLWorld_LI*, where HI stands for relatively higher illicit rate and LI stands for lower illicit rate. The statistics of the datasets are presented in Table 4.

**Baselines. SGM** refers to the centralized scatter-gather mining method described in Section 2.1. A line of research leverages graph

**Table 1: Experiments on AMLSim and AMLWorld datasets. "-" represents that the metric is unsuitable for the method.**

| | *AMLSim_bal* | | | | | *AMLSim_unb* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Methods** | ACC | Precision | Recall | F1-score | AUC | ACC | Precision | Recall | F1-score | AUC |
| SGM | 0.9761 | 0.8627 | 0.9047 | 0.8832 | **0.9743** | 0.9805 | 0.8665 | 0.9678 | 0.9144 | 0.9876 |
| GIN [7, 32] | 0.9497±0.0021 | 0.8397±0.0096 | 0.8992±0.0033 | 0.8684±0.0047 | 0.9301±0.0016 | 0.8978±0.0064 | 0.6926±0.0159 | 0.9045±0.0034 | 0.7844±0.0109 | 0.9003±0.0049 |
| GAT [24] | 0.8332±0.0123 | 0.5285±0.0201 | 0.9173±0.0021 | 0.6704±0.0159 | 0.8657±0.0071 | 0.8235±0.0181 | 0.5424±0.0288 | 0.9251±0.0035 | 0.6835±0.0225 | 0.8611±0.0113 |
| PNA [25] | 0.9533±0.0017 | **0.8508±0.0078** | 0.9061±0.0018 | **0.8776±0.0039** | 0.9351±0.0011 | 0.9177±0.0043 | **0.7470±0.0125** | 0.9066±0.0022 | **0.8190±0.0079** | 0.9136±0.0032 |
| LaundroGraph [4] | 0.936±0.0014 | 0.7870±0.0048 | 0.8961±0.0034 | 0.8380±0.0031 | 0.9206±0.0018 | 0.9136±0.0043 | 0.7350±0.0126 | 0.9070±0.0077 | 0.812±0.0077 | 0.9112±0.0029 |
| MultiGIN [5] | **0.9827±0.0003** | **0.9949±0.001** | **0.9108±0.0016** | **0.951±0.0008** | **0.9549±0.0008** | **0.9809±0.0005** | **0.9955±0.0012** | **0.9110±0.0018** | **0.9514±0.0012** | **0.955±0.0009** |
| Prob-CSGM | **0.9701±0.0022** | 0.8358 ±0.0185 | 0.872±0.0074 | 0.8534±0.0095 | - | 0.9392±0.0019 | 0.6576±0.0049 | 0.9062±0.0161 | 0.7621 ±0.0086 | - |
| Sim-CSGM | 0.9699±0.0007 | 0.8005±0.0086 | **0.9296±0.0091** | 0.8601±0.0019 | 0.9519±0.0038 | **0.9427±0.0002** | 0.655±0.0009 | **0.9868±0.0010** | 0.7874±0.0006 | **0.9621±0.0004** |
| | *AMLWorld_HI* | | | | | *AMLWorld_LI* | | | | |
| **Methods** | ACC | Precision | Recall | F1-score | AUC | ACC | Precision | Recall | F1-score | AUC |
| SGM | 0.9992 | 0.5187 | 0.6501 | 0.5770 | 0.8250 | 0.9989 | 0.0314 | 0.1765 | 0.0533 | 0.5878 |
| GIN [7, 32] | 0.9984±0.0004 | 0.2938±0.0781 | 0.5526±0.0892 | 0.3811±0.0828 | 0.7757±0.0447 | 0.9997±0.0001 | 0.1791±0.0350 | 0.1647±0.0738 | 0.1598±0.0474 | 0.5823±0.0369 |
| GAT [24] | 0.9992±0.0001 | 0.5572±0.1188 | 0.2143±0.0136 | 0.3081±0.0326 | 0.6071±0.0068 | 0.998 | 0.0 | 0.0 | 0.0 | 0.5 |
| PNA [25] | 0.9985±0.0001 | 0.3565±0.0208 | **0.938±0.0097** | 0.5165±0.0234 | **0.9683±0.0049** | 0.9997±0.0001 | 0.3321±0.0052 | **0.8873±0.0069** | **0.4833±0.0065** | **0.9435±0.0035** |
| LaundroGraph [4] | 0.9992±0.0001 | 0.5412±0.0840 | 0.6193±0.0613 | 0.5710±0.0299 | 0.8094±0.0306 | 0.9998±0.0043 | 0.3846±0.0126 | 0.0490±0.0077 | 0.0870±0.0077 | 0.5245±0.0029 |
| MultiGIN [5] | 0.9996±0.0002 | 0.6945±0.0959 | **0.9366±0.0173** | **0.7943±0.0658** | **0.9681±0.0086** | 0.9996±0.0001 | 0.1746±0.0365 | 0.3353±0.2252 | 0.2104±0.1101 | 0.6675±0.1125 |
| Prob-CSGM | **0.9996±0.0001** | **0.8747±0.0242** | 0.6413±0.0643 | 0.7392 ±0.0499 | - | **0.9998±0.0001** | **0.4370±0.0684** | 0.3529±0.1038 | 0.3878 ±0.086 | - |
| Sim-CSGM | **0.9997±0.0001** | **0.9347±0.0191** | 0.6837±0.0136 | **0.7897±0.0128** | 0.8419±0.0068 | **0.9999 ±0.0009** | **0.9211±0.0008** | **0.6863±0.0001** | **0.7865±0.0041** | **0.8431±0.0002** |

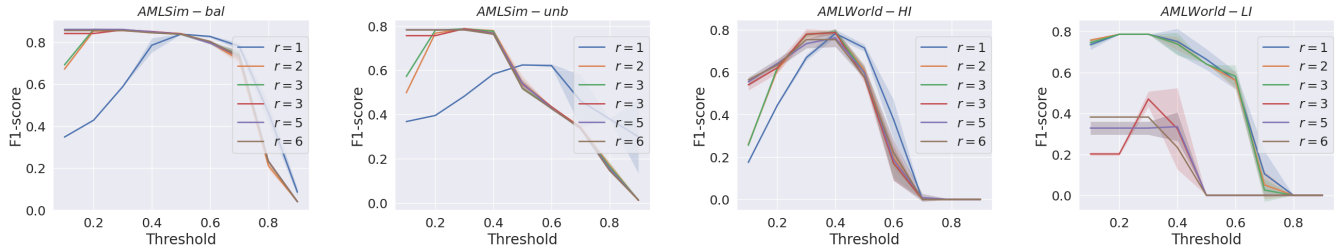



**Figure 6: Experiments for the similarity-based method with the four synthetic datasets.**

neural networks (GNNs) for identifying money laundering transactions. **GIN**[7, 32], **GAT**[24], and **PNA**[25] are commonly used GNN models for general graph classification tasks. Two additional studies propose GNNs specifically tailored for money laundering detection. **LaundroGraph**[4] introduces a self-supervised graph representation learning method aimed at detecting money laundering. **MultiGIN**[5], incorporates a range of adaptations, including multigraph port numbering, ego IDs, and reverse message passing, to enhance GNNs' ability to detect various patterns of illicit activities.

## 5.2 Experiments on Alipay-ECB.

**Data process.** Performing experiments on the Alipay-ECB dataset is challenging due to the massive volume of transaction records. To facilitate the implementation of our methods, we process the data as follows:

- **Account Treatment:** A user may link deposits or credit cards from different banks to their Alipay account. As a result, numerous transactions occur between Alipay and the user's cards (e.g., through withdrawal services). To facilitate the tracing of fund flows between different banks, we treat each card as a separate account, even if they belong to the same user.
- **Transaction Simplification:** Transactions between two accounts may occur multiple times. However, as we mainly focus on set of transaction, we consolidate these transactions into a single entry. This is different from MultiGIN [5], which treats transactions between the same accounts as distinct entities.
- **Transaction Amount Filtering:** Given that money laundering often involves large sums, we filter out transactions with small amounts (around ¥100 in our experiments).

After applying these steps, we obtain a transaction graph with 48.95 million accounts and 34.45 million transactions. Detailed statistics are presented in Table 2.

**Table 2: Statistics of Alipay-ECB after processing.**

| Accounts | | | Transactions | | | |
|---|---|---|---|---|---|---|
| Alipay | ECB | Others | Alipay → Alipay | ECB → ECB | Alipay→ ECB | ECB→ Alipay |
| 23.46M | 3.99M | 21.50M | 30.84M | 5.51M | 0.25M | 1.65M |

**Results.** We experiment with similarity-based methods and set the threshold to 0.1. Examples of detected groups are presented in Figure 8. We randomly selected 100 detected groups and evaluated them based on whether the accounts in each group had been reported as illegal [1].

Among the 100 detected groups, 59 were identified as money laundering groups, with more than half of their accounts recognized as illicit in actual business operations. Two groups were identified

[1]For reasons related to corporate confidentiality and data safety, we regret that we cannot disclose the exact figures of detected groups.

as non-money laundering, while the remaining 39 groups contained at least one illicit account. The results demonstrate that our method can effectively detect money laundering groups. With the group-level information, our methods would assist in uncovering previously undiscovered illicit accounts.

### 5.3 Experiments on synthetic datasets.

We experiment with Prob-CSGM and Sim-CSGM on four synthetic datasets and compare them with all five baselines. We set the number of hash functions $m$ used in MinHash to 100, and $r$ to 7 for Prob-CSGM. For Sim-CSGM, $m$ is set to 100, $r = 2$, and the threshold values used in *AMLSim* and $AMLWorld$ are set to 0.4. The size of the Bloom filter is 500,000 bits for AMLSim and 3,000,000 bits for AMLWorld, resulting in a false positive probability of approximately 0.01.

The results are shown in Table 1. On the two AMLSim datasets, our methods have a performance comparable to SGM. Sim-CSGM, in particular, outperforms the centralized method in terms of recall, indicating that it is more effective at identifying abnormal nodes comprehensively. On the AMLWorld datasets, SGM exhibits low precision, with even poorer performance on AMLWorld-LI. This is because SGM tends to identify small transaction groups as money laundering groups, which is normal in AMLWorld-HI. However, Sim-CSGM can still identify money laundering groups, demonstrating its robustness.

Compared to GNN-based methods, our approach has a comparable performance on the AMLSim dataset, which achieves the best results with both precision and recall rates exceeding 90%. However, when the proportion of illicit transactions is exceptionally low, such as in the AMLWorld-HI dataset, MultiGIN suffers from low precision, leading to a high false positive rate. In contrast, our methods maintain strong performance on the AMLWorld datasets, highlighting the generalizability of our approach.

### 5.4 Ablation study.

To explore the impact of different parameters on the performance of our methods, we conducted experiments by varying the number of hash functions used in MinHash $m$, the number of rows $r$ in each band as well as the threshold.

Here, we mainly focus on Sim-CSGM. . We vary the threshold from 0.2 to 0.6 and observe the change of F1-score with different $r$. The results are depicted in Figure 6. It shows that the F1-score when $r > 1$ performs better than when $r = 1$, showing the effectiveness of the banding technique in the similarity-based method. Furthermore, when $r = 1$, the method prefers a higher threshold, illustrating that repeated elements in a band lead to an overestimation of similarity when using the bloom filter.

Additionally, experiments in Appendix A.3 show that the banding technique could significantly reduce the number of repeated elements. We also evaluate the efficiency of our methods in terms of the communication costs as well as the running time in Appendix A.4. The results show that our methods take only a few minutes.

## 6 RELATED WORKS

The term money laundering was first used at the beginning of the 20th Century to label the operations that in some way intended to legalize the income derived from illicit activity, thus facilitating their entry into the monetary flow of the economy [22]. Since then, numerous methods have been proposed to identify money laundering activities [6, 10, 13, 17–19, 33]. Rule-based approaches were first widely used in the early days [13, 17]. Rajput et al. [17] propose an ontology-based expert system to detect suspicious transactions, and Michalak et al. [13] propose a method that integrates the fuzzing method and decision rules to detect suspicious transactions. Although easy to deploy, rule-based methods can easily be evaded by fraudsters.

With the popularity of machine learning, learning-based methods have become an emergency. Tang et al. [21] propose to use the support vector machine method (SVM) to detect unusual behaviors in transactions. Lv et al. [12] judge whether the capital flow is involved in money laundering activities using RBF neural networks calculated from time to time. Paula et al. [16] also show some success for AML by using deep neural networks. However, these methods detect money laundering activities in a supervised manner, suffering from highly skewed labels and limited adaptability. Recently, Li et al. [11] propose a metric to evaluate the anomalousness of a subgraph induced by a subset of nodes and propose an algorithm to find subsets that maximize the metric. The subsets are treated as suspicious money laundering groups.

Graphs have the advantage of better characterizing the association between objects. Many graph-based anomaly detection techniques have been developed for discovering structural anomalies. Zhang et al. [33] use financial transaction networks and community detection algorithms to find money laundering groups. Cardoso et al. [4] introduces a self-supervised graph representation learning method aimed at detecting money laundering. Recently, *Bni* et al. [5], incorporates a range of adaptations, including multigraph port numbering, ego IDs, and reverse message passing, to enhance GNNs' ability to detect various patterns of illicit activities.

Despite the advance of all those methods, they work based on the prerequisite that the transaction graph is centralized, while in practice, money laundering activities span across multiple institutions s.t. the transaction subgraph within one institution appears to be normal. Our methods make the *first* steps towards collaborative anti-money laundering among institutions without exposing the transaction graphs.

## 7 CONCLUSION

In this work, we propose the *first* algorithm enabling collaborative anti-money laundering (AML) among institutions while preserving the privacy of their transaction graphs. We employ LSH [30] and bloom filters [27] to reduce communication costs and enhance efficiency. Experimental evaluations on two synthetic datasets demonstrate the effectiveness and efficiency of the proposed algorithm. In future work, we will attempt to deploy the algorithm in real-world industrial settings to evaluate its effectiveness with realistic data. Moreover, we will enhance the algorithm to address intricate money laundering scenarios involving more institutions and more complex transaction graphs.

# REFERENCES


[1] 2024. alipay. https://www.alipay.com/.

[2] 2024. E-Commerce Bank. https://www.mybank.cn/.

[3] Erik Altman, Jovan Blanuša, Luc Von Niederhäusern, Béni Egressy, Andreea Anghel, and Kubilay Atasu. 2024. Realistic synthetic financial transactions for anti-money laundering models. *Advances in Neural Information Processing Systems* 36 (2024).

[4] Mário Cardoso, Pedro Saleiro, and Pedro Bizarro. 2022. LaundroGraph: Self-supervised graph representation learning for anti-money laundering. In *Proceedings of the Third ACM International Conference on AI in Finance*. 130–138.

[5] Béni Egressy, Luc Von Niederhäusern, Jovan Blanuša, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. 2024. Provably Powerful Graph Neural Networks for Directed Multigraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 11838–11846.

[6] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 895–904.

[7] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* (2019).

[8] Kaggle. 2024. IBM Transactions for Anti Money Laundering (AML). https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml/data.

[9] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. 2017. Practical multi-party private set intersection from symmetric-key techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1257–1272.

[10] Nhien An Le Khac and M-Tahar Kechadi. 2010. Application of data mining for anti-money laundering detection: A case study. In *2010 IEEE international conference on data mining workshops*. IEEE, 577–584.

[11] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. 2020. Flowscope: Spotting money laundering based on graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 4731–4738.

[12] Lin-Tao Lv, Na Ji, and Jiu-Long Zhang. 2008. A RBF neural network model for anti-money laundering. In *2008 International conference on wavelet analysis and pattern recognition*, Vol. 1. IEEE, 209–215.

[13] Krzysztof Michalak and Jerzy Korczak. 2011. Graph mining approach to suspicious transaction detection. In *2011 Federated conference on computer science and information systems (FedCSIS)*. IEEE, 69–75.

[14] United Nations. 2020. Tax abuse, money laundering and corruption plague global finance.

[15] United Nations. 2024. Money Laundering. https://www.unodc.org/unodc/en/money-laundering/overview.html.

[16] Ebberth L Paula, Marcelo Ladeira, Rommel N Carvalho, and Thiago Marzagao. 2016. Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In *2016 15th ieee international conference on machine learning and applications (icmla)*. IEEE, 954–960.

[17] Quratulain Rajput, Nida Sadaf Khan, Asma Larik, and Sajjad Haider. 2014. Ontology based expert-system for suspicious transactions detection. *Computer and Information Science* 7, 1 (2014), 103.

[18] Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Alaa Yagoub, and Aijun An. 2016. A new algorithm for money laundering detection based on structural similarity. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 1–7.

[19] Michele Starnini, Charalampos E Tsourakakis, Maryam Zamanipour, André Panisson, Walter Allasia, Marco Fornasiero, Laura Li Puma, Valeria Ricci, Silvia Ronchiadin, Angela Ugrinoska, et al. 2021. Smurf-based anti-money laundering in time-evolving transaction networks. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV 21*. Springer, 171–186.

[20] Toyotaro Suzumura and Hiroki Kanezashi. 2021. Anti-Money Laundering Datasets: InPlusLab Anti-Money Laundering DataDatasets. http://github.com/IBM/AMLSim/.

[21] Jun Tang and Jian Yin. 2005. Developing an intelligent data discriminating system of anti-money laundering based on SVM. In *2005 International conference on machine learning and cybernetics*, Vol. 6. IEEE, 3453–3457.

[22] Rodolfo Uribe. [n. d.]. Changing Paradigms on Money Laundering. http://www.cicad.oas.org/oid/new/information/observer/observer2_2003/mlparadigms.pdf.

[23] Atif Usman, Nasir Naveed, and Saima Munawar. 2023. Intelligent Anti-Money Laundering Fraud Control Using Graph-Based Machine Learning Model for the Financial Domain. *Journal of Cases on Information Technology* 25, 1 (2023).

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[25] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR (Poster)* 2, 3 (2019), 4.

[26] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. 2018. Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076* (2018).

[27] Wikipedia. 2023. Bloom Filter. https://en.wikipedia.org/wiki/Bloom_filter.

[28] Wikipedia. 2023. Know Your Customer. https://en.wikipedia.org/wiki/Know_your_customer.

[29] Wikipedia. 2023. MinHash. https://en.wikipedia.org/wiki/MinHash.

[30] Wikipedia. 2024. Locality-sensitive hashing. https://en.wikipedia.org/wiki/Locality-sensitive_hashing.

[31] Wikipedia. 2024. MD5. https://en.wikipedia.org/wiki/MD5.

[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[33] Zhongfei Zhang, John J Salerno, and Philip S Yu. 2003. Applying data mining in investigating money laundering crimes. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 747–752.

# A APPENDIX

**Table 3: Summary of notations**

| Notation | Description |
|---|---|
| $\mathcal{G}$ | transaction graph |
| $\mathcal{G}'$ | transaction graph with inversed direction |
| $m$ | number of hash functions used in MinHash |
| $r$ | number of rows of each band |
| $K$ | number of bloom filters |
| $S$ | the set of cross-institution transactions find with $\mathcal{G}$ |
| $D$ | the set of cross-institution transactions find with $\mathcal{G}'$ |
| $\mathcal{S}, \mathcal{D}$ | list of sets $S$ and $D$ |
| $M$ | Signature matrix |
| $B$ | Banding matrix |

## A.1 Theoretical Analysis

Theorem 1 (restated). *Suppose that $X_1, ..., X_N$ are a sequence of real values with $0 \le X_N \le ... \le X_1 \le 1$. Then $\forall \varepsilon > 0$, there $\exists \delta$, s.t. when $r > \delta$,*

$$(\sum_{i=1}^{N} X_1^r)^{1/r} - X_1 \le \varepsilon.$$

Proof. We have

$$\begin{aligned}
&(\sum_{i=1}^{N} X_1^r)^{1/r} - X_1 \\
=&\left(x_1^r\left(1+\sum_{i=1}^{N-1}\left(\frac{X_i}{X_1}\right)^r\right)\right)^{1/r} - X_1 \\
=&X_1\left(1+\sum_{i=1}^{N-1}\left(\frac{X_i}{X_1}\right)^r\right)^{1/r} - X_1 \\
=&X_1\left(\left(1+\sum_{i=1}^{N-1}\left(\frac{X_i}{X_1}\right)^r\right)^{1/r} - 1\right)
\end{aligned}$$

To prove the inequality, we only need to prove

$$\left(1+\sum_{i=1}^{N-1}\left(\frac{X_i}{X_1}\right)^r\right)^{1/r} < \frac{\varepsilon}{X_1} + 1$$

It is easy to prove that

$$\begin{aligned}
\left(1+\sum_{i=1}^{N-1}\left(\frac{X_i}{X_1}\right)^r\right)^{1/r} &\le \left(1+(N-1)\left(\frac{X_2}{X_1}\right)^r\right)^{1/r} \\
&< 1+(N-1)\left(\frac{X_2}{X_1}\right)^r
\end{aligned}$$

When $r > \log_p(\frac{\varepsilon}{X_1(N-1)})$, we have

$$1+(N-1)\left(\frac{X_2}{X_1}\right)^r < \frac{\varepsilon}{X_1} + 1,$$

where $p = X2/X1$

□

## A.2 Dataset Statistics

**Alipay-ECB.** Alipay Mobile Payment [1] is the world's largest mobile payment platform, allowing users to pay for a wide range of daily needs, including money transfers, online shopping, salary deposits, investments, and more. As of June 2020, Alipay serves over 1.3 billion users and 80 million merchants [**?** ], making it an invaluable resource for studying money laundering activities that may be concealed within its vast volume of transaction records. Meanwhile, E-Commerce Bank, operating entirely online, serves tens of millions of users and merchants across China, facilitating numerous financial transactions between Alipay and E-Commerce Bank daily. The AlipayECB dataset captures these transactions, with the majority of records originating from Alipay. These include transactions between Alipay users as well as between users and various bank accounts, with ESB being one of the many banks involved.

**Time span.** The AlipayECB dataset is constructed using transactions that occurred on Alipay and E-Commerce Bank (ECB) within a single day. Unlike synthetic datasets such as AMLSim [20] and AMLWorld [3], where transactions within a single money laundering group can span several days [8], money laundering transactions on digital platforms tend to occur rapidly. Funds are moved in and out quickly to minimize the risk of losses due to account monitoring and censorship. Based on this observation, we focus exclusively on transactions occurring within a single day.

**Examples of discovered subgraphs** Figure 8 presents examples of detected groups. Money laundering groups are identified when more than half of their accounts are classified as illicit. Grey groups are those in which only a small number of accounts have been reported for suspected money laundering activities. The normal group is associated with school financial collections.

**Machine Specs and code.** The experiments on synthetic datasets were conducted on a single machine using Python 3. The experiments on Alipay-ECB were carried out in Java on a cluster of 20 machines.

## A.3 The impact of number of rows $r$

To further investigate the impact of $r$ on estimating similarity in Sim-CSGM, we plot the frequency histogram of repeated elements within a band, as shown in Figure 7. The x-axis represents the number of repetitions, while the y-axis indicates the number of sets that share repeated elements with others. As the number of repeated sets increases, more bias is introduced into the similarity estimation. The results show that most sets differ from others (repeat = 1), but many still share repeated elements. However, the banding technique can significantly reduce the number of repeated elements.

## A.4 Efficiency.

We evaluate the efficiency of our algorithm in terms of the communication costs among institutions as well as the running time.

**Communication costs.** The communication costs are primarily induced by the transmission of Bloom filters between institutions. Specifically, each institution needs to transmit its Bloom filters to another institution, with each filter containing values within a band. In our experiments, the Bloom filter size is 500,000 for AMLSim and 3,000,000 for AMLWorld, corresponding to approximately 61.04 KiB and 366.21 KiB, respectively. In our experiments, we adopt 100

**Table 4: Statistics of datasets. $|\mathcal{V}|$ denotes the total number of accounts, $|\mathcal{E}|$ denote the total number of transactions, and $|\mathcal{V}_p|$ as well as $|\mathcal{E}_p|$, where $p \in [A, B]$, represent the number of accounts and transactions owned by $\mathcal{P}_p$. IR represents an illicit ratio of accounts.**

| Dataset | $\lvert\mathcal{V}\rvert$ | $\lvert\mathcal{E}\rvert$ | $\lvert\mathcal{V}_A\rvert$ | $\lvert\mathcal{E}_A\rvert$ | $\lvert\mathcal{V}_B\rvert$ | $\lvert\mathcal{E}_B\rvert$ | IR |
|---|---|---|---|---|---|---|---|
| *AMLSim_bal* | 100K | 1 957 005 | 50K | 1 653 870 | 50K | 1 364 455 | 9.97% |
| *AMLSim_unb* | 100K | 1 959 234 | 75K | 1 844 664 | 25K | 919 211 | 10.75% |
| *AMLWorld_HI* | 515 020 | 5 073 772 | 257 799 | 3 718 584 | 257 221 | 3 592 358 | 0.070% |
| *AMLWorld_LI* | 705 861 | 6 920 656 | 352 189 | 5 096 798 | 353 672 | 4 880 541 | 0.014% |
| *Alipay-ECB* | 245M | 336M | 225M | 309M | 19M | 26M | - |

**Algorithm 3:** Cross-institution Transaction Set Discovery

**Input:** node $i$, directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\vec{c} \in \mathcal{X}$ represents whether the transaction is an cross-institution transaction, depth $K$

**Output:** Set of cross-institution transactions that scattered from $i$

```
Initialize set S = ∅
Denote N_i = {j|(i, j) ∈ E}, ∀i ∈ V
Initialize set Q = {i}
for k = {1, ..., K} do
    Initialize set Z = ∅
    for v ∈ Q do
        for j ∈ N_v do
            if c_{v→j} = 1 then
                S.append((v, j))
            else
                Z.append(j)
            end
        end
    end
    if Z == ∅ then
        break
    else
        Q = Z
    end
end
return S
```

Bloom filters, resulting in about 13 MiB and 75 MiB of data transfer per institution.

**Running time.**

We report the running time of applying our methods on the two datasets in Table **??**. The running time for each stage of during the execution of the methods is presented, including discovering sets of cross-institution transactions, performing MinHash functions on all sets, inserting bands into Bloom filters, and testing the existence of elements in Bloom filters. On the two AMLSim datasets, our methods take only a few seconds. Even on the larger AMLWorld datasets, the methods still take only a few minutes, demonstrating its efficiency. The MinHash calculation process is the most time-consuming, while the membership testing stage is highly efficient, requiring just a few seconds.

## A.5 Additional Algorithms

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

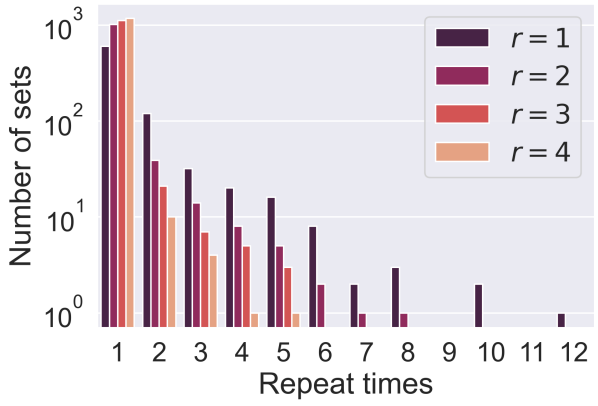
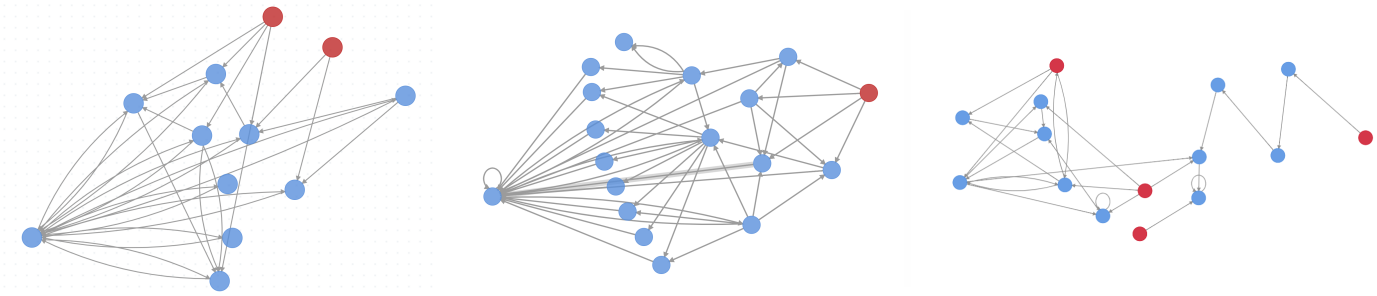


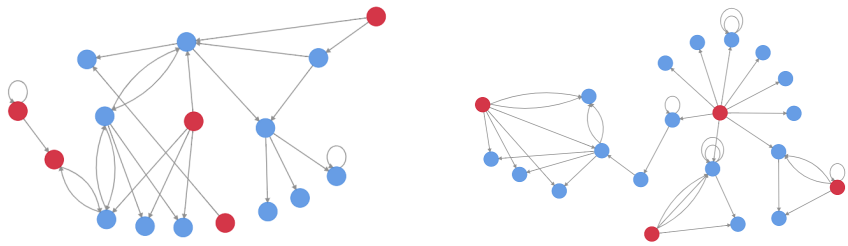

**Figure 7: The frequency histogram of repeated elements in a band.**

**Table 5: Running time in seconds of Sim-CSGM. Set discovery represents the process of discovering sets of cross-institution transactions, minhash presents performing minhash functions to all sets, inserting represents inserting bands to bloom filters, and membership testing represents testing the existence of elements in bloom filters..**

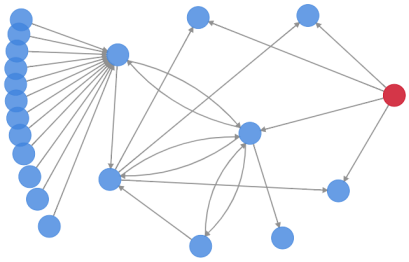
| Dataset | Inst. | Set Discovery | Minhash | Inserting | Membership Testing | Total |
|---|---|---|---|---|---|---|
| *AMLSim-bal* | $\mathcal{P}_A$ | 1.74 | 3.49 | 1.80 | 0.41 | 7.44 |
| | $\mathcal{P}_B$ | 2.26 | 7.78 | 2.10 | 0.63 | 12.77 |
| *AMLSim-unb* | $\mathcal{P}_A$ | 4.23 | 4.83 | 2.84 | 0.82 | 12.72 |
| | $\mathcal{P}_B$ | 1.11 | 5.80 | 5.46 | 0.96 | 13.33 |
| *AMLWorld-HI* | $\mathcal{P}_A$ | 7.67 | 186.45 | 10.57 | 2.93 | 207.62 |
| | $\mathcal{P}_B$ | 1.27 | 8.70 | 6.91 | 2.56 | 19.44 |
| *AMLWorld-LI* | $\mathcal{P}_A$ | 7.81 | 184.1 | 10.71 | 2.91 | 205.53 |
| | $\mathcal{P}_B$ | 1.29 | 8.90 | 6.99 | 2.52 | 19.70 |



**(a) Money laundering groups**



**(b) Grey groups**



**(c) Normal groups**

**Figure 8: Examples of money laundering groups detected by CSGM, with colors indicating different institutions. The blue nodes represent accounts from Alipay and the red nodes represent accounts from ECB.**