Privacy-preserving Anti-Money Laundering using Secure Multi-Party Computation

Marie Beth van Egmond[1], Vincent Dunning[1], Stefan van den Berg[1], Thomas Rooijakkers[1], Alex Sangers[1], Ton Poppe[2], and Jan Veldsink[3]

[1] Netherlands Organisation for Applied Scientific Research (TNO), The Netherlands
[2] ABN AMRO, The Netherlands
[3] Rabobank, The Netherlands

Abstract. Money laundering is a serious financial crime where criminals aim to conceal the illegal source of their money via a series of transactions. Although banks have an obligation to monitor transactions, it is difficult to track these illicit money flows since they typically span over multiple banks, which cannot share this information due to privacy concerns. We present *secure risk propagation*, a novel efficient algorithm for money laundering detection across banks without violating privacy concerns. In this algorithm, each account is assigned a *risk score*, which is then propagated through the transaction network. In this article we present two results. Firstly, using data from a large Dutch bank, we show that it is possible to detect unusual activity using this model, with cash ratio as the risk score. With a recall of 20%, the precision improves from 15% to 40% by propagating the risk scores, reducing the number of false positives significantly. Secondly, we present a privacy-preserving solution for securely performing risk propagation over a joint, inter-bank transaction network. To achieve this, we use Secure Multi-Party Computation (MPC) techniques, which are particularly well-suited for the risk propagation algorithm due to its structural simplicity. We also show that the running time of this secure variant scales linearly in the amount of accounts and transactions. For $200,000$ transactions, two iterations of the secure algorithm between three virtual parties, run within three hours on a consumer-grade server.

Keywords: financial crime · money laundering · multi-party computation · privacy · risk propagation

1 Introduction

Financial Economic Crime is a serious crime with a huge social and economic impact on our society. In 2018 alone, an estimated 5.8 trillion dollars (6.5% of global GDP) worth of financial crime was conducted [9]. Money laundering is, after fraud, the second largest economic crime, as it often co-occurs with other type of crimes like drug trafficking, human trafficking and terrorist financing [6]. There are multiple definitions of money laundering as stated by [2]. In this paper we use the definition from the European Union Law: "Money laundering

is the process by which criminals conceal the illegal origin of their property or income" [12].

Money laundering can come in many forms, and new ways of money laundering are invented by criminals continuously. The general pattern can be described by three phases: placement, layering, and integration [14]. During the *placement* phase, the cash from illicit activities is brought into the financial system, for example via multiple small cash deposits. The *layering* phase consists of a series of transactions aimed at disguising the source of funds. And in the final *integration* phase, the money launderers return the illicit money back to themselves in a way that looks legitimate.

These phases are searched for in transaction monitoring. The objective of transaction monitoring is to detect unusual transactions that might be related to money laundering and/or terrorist financing. The anti-money laundering (AML) directives of the European Union, implemented in the Dutch WWTF (law for preventing money laundering and terrorism financing) obliges financial institutions to implement a transaction monitoring process [19].

A transaction monitoring process starts with the systematic identification of the integrity risks towards money laundering, and how these risks are mitigated. This mitigation is often performed via a transaction monitoring system, but can also be performed by manual inspection at the front desk. The automated transaction monitoring system flags unusual behavior referred to as 'alerts'. The alerts are manually investigated and in case of a true hit in the context of WWFT, a suspicious activity report is reported to the Financial Intelligence Unit. This organization takes further action. Conventional transaction monitoring systems are rule-based and cannot always capture the complexity related to the transaction behavior of customers. This results in a large false positive rate of up to 95%, which means many costly investigations have to be performed that do not lead to detecting criminal activity. The usage of machine learning can decrease the number of false positives and thereby the overall performance of the system [13].

The general money laundering pattern of placement, layering and integration shows that this involves a series of transactions with multiple entities, across banks and jurisdictions. This motivates a network view with the entities as nodes and transactions as edges. However, banks only see their own part of the transaction graphs, only related to their own customers' transactions. Understandably, due to privacy constraints, regulations and business sensitivity, banks cannot simply share this information with each other. Banks are therefore investigating collaboration whilst taking privacy concerns into account. One possible solution is to use a Trusted Third Party, that gathers all data, performs the analysis and reports the results back to the parties involved. However, such initiatives can still be confronted with privacy concerns. A decentralized alternative for a TTP is given by applying Secure Multi-Party Computation (MPC). MPC is a set of cryptographic techniques that essentially mimic the TTP by means of a protocol the parties can jointly execute while staying in full control of their own data. This removes the need to trust anyone with their data. MPC techniques enable

the use of data from multiple sources to obtain shared knowledge without actually sharing the data with each other. MPC is a Privacy Enhancing Technology (PET), which have been recognized as a promising technique in the fight against financial crime by the Future of Financial Intelligence Sharing programme. Their white paper [17] describes ten case studies of using PETs in the fight against financial crime.

Related Work Many data-driven techniques have been applied to detect money laundering, varying from traditional rule-based models to various machine learning models [1, 24]. Only a small subset of those concern algorithms that exploit the graph structure of bank transactions [18]. For example, social network analyses have been applied successfully on transaction data to prevent and detect money laundering [7, 11]. However, such methods are primarily aimed to be applied on transaction data of a single bank and typically not suitable to be extended to a distributed network (over multiple banks) using MPC due to their complexity.

An overview of the use of PETs in finance in general can be found in the Systemization of Knowledge by Baum et al. [5]. An earlier version of our work [28] and the work of Zand et al. [31] are mentioned as the main bodies of work in the field of cryptography and AML. In the latter research, a money-laundering detection system is presented that includes one or more auditors, who analyse transactions from multiple banks, via some confidentialy-preserving cryptographic protocol. The suspicious activity reports (SARs) that follow from this audit are then secret shared to the banks, that can finally recover the SAR together when they collectively decide on doing so. Note that the difference with our approach is that we do not need a (trusted) third party.

Some real-world use cases for the use of PETs in the fight against financial crime are mentioned in a paper by Future of Financial Intelligence Sharing (FFIS) [17]. One of these examples focuses on network data coming from the Australian Transaction Reports and Analysis Centre (ASTRAC) [15]. The algorithm is based on the observation that the nodes associated to the layering phase in money laundering show similarities in terms of structure and information. The similarities of all nodes in the transaction graph are compared pairwise, in order to identify clusters of similar nodes. This way money laundering patterns can potentially be detected. de Perthuis et al. [10] also look at similarities within a transaction network. They present a secure way, using functional encryption, of an idea presented by Soltani et al. [26]. They present a framework in which reduced transaction data can be scanned to find pairs of transactions with common attributes and behaviours that are potentially involved in money laundering activities. It then applies a clustering method to detect potential money laundering groups. Using the method of de Perthuis, this can be done between two banks without sharing their part of the transaction network. However, both AUSTRAC and de Perthuis use a pairwise comparison approach, which scales quadratically in the number of accounts and requires performing non-linear comparisons. This is very costly in terms of communication in an MPC protocol.

In an article by Suzumura et al. [27], federated machine learning has been used on transaction graphs. There a prediction model is trained for suspicious money laundering efforts by using local and global graph features. Aly et al. [3] consider the shortest path and max-flow algorithm using MPC. The PageRank algorithm [22] was implemented in a multi-party setting by Cozzo et al. [8] using an active security approach. There, the computation can be outsourced to other parties. However, the computation- and communication complexity of all of these solutions renders them infeasible for large transaction graphs and thus for a wider use in detecting financial crime.

Several frameworks for efficiently computing graph algorithms have been proposed in [4, 21]. By outsourcing the computation to a set of three other parties of which at least two are honest, a tailored MPC solution can be used to overcome performance issues at the cost of having a weaker security model where the banks are furthermore no longer in complete control of their own customer data. This is undesirable in practice.

Finally, Sangers et al. [25] did develop a scalable solution (linear in the number of nodes) for computing PageRank securely while keeping in control of your own data, using additive homomorphic encryption. They exploit the fact that each party knows its own transaction graph, enabling local computations with private values. While PageRank is a relevant metric used for detecting transactional fraud, it is unclear if PageRank can contribute in detecting money laundering patterns. However, the concept of exploiting local knowledge presented in their work serves as a basis for our work.

Contributions

- We present a novel algorithm called *Risk Propagation* for performing anti-money laundering by analyzing transaction graphs. This is a relatively new and promising approach for banks to do AML. Our algorithm is agnostic towards the type of indicators of risk that are of interest and is thus flexible to be used for analyzing various types of risk. The design of this algorithm was devised with the goal of privacy-preserving analyses in mind.
- We present an efficient solution for performing our risk propagation algorithm in a privacy-preserving way. For this, we present a scalable solution using *additively homomorphic encryption*. This enables collaborative anti-money laundering across financial institutions on transaction graphs of millions of accounts. With this solution, we address the problem that the performance of such transaction graph analyses is typically hampered by the limited view a single bank has on the transaction graph. In our approach, the parties stay in complete control of their own data in contrast to other works that tackle similar algorithms [4, 21] and do not need any external party to perform the computations for them.
- We have evaluated our solution on real transaction data from a large, Dutch bank and using cash ratios as indicators to show the potential of our algorithm as a feature to improve AML algorithms.

Outline The rest of this article is structured as follows. In Section 2, the risk propagation algorithm is presented. Section 3 demonstrates how one can apply additive homomorphic encryption to the risk propagation algorithm, so that it can be used securely in a multi-party setting. In the first part of Section 4, we present results of experiments of the standard risk propagation on real bank data to measure its predictive power. In the second part of Section 4, we present the results of experiments performed to measure the scalability of the secure, privacy-preserving multi-party variant. Finally, in Section 5 we conclude this research, discuss the results of this work and give pointers towards future research. An overview of the notation can be found in Table 1.

2 Risk Propagation Algorithm

The approach of the risk propagation algorithm is to update risk scores of every bank account, by using risk scores of accounts that it receives money from, in a weighted manner. In other words, if an account receives money from an account with a higher (or lower) risk score, its updated risk score becomes higher (or lower). This way, the layering phase of money laundering is simulated, and accounts involved could possibly be detected using the updated risk scores. Also, multiple iterations are used to mimic the depth of the layering phase, so that illicit money flows can be followed. In this section, we present the details of the risk propagation algorithm.

2.1 Transaction data to a static graph

The input of the risk propagation algorithm is transaction data. These data consists of the accounts of a bank and all transactions between these accounts over a certain period of time. We aggregate the transactions that have been made in this time period into two amounts between every account i and j. The first amount $A_{i,j}$ is the total amount of all transactions sent from account i to j, and the second amount $A_{j,i}$ is the total amount from j to i. These amounts could also be zero when there is no transaction. In the upcoming sections, we consider the accounts to be nodes and the amounts to be the weights of directed edges, forming weight matrix A, resulting in a simple directed graph. Furthermore, every account has a previously defined risk score. In the experiments in Chapter 4 we chose cash ratio as a risk score. Other choices could be based on high-risk geographies or the use of cryptocurrencies. In the upcoming sections we consider risk scores more generally as node attributes.

Note that ideally we would use the risk propagation algorithm in real time, updating every risk score when there is a new transaction. However, we have the aim of applying the algorithm in a secure multi-party setting, as described in the next section. To reduce complexity, we need to perform the algorithm on a static graph. However, in the experiments we split up into four sequential time periods, to mimic real time propagation. Here the updated risk scores over a network of one time period are used as input for the network of the next time period. The smaller these time periods are, the closer it gets to a real time propagation.

2.2 Input Data

The input of the algorithm is a directed graph $G(V, E)$, consisting of nodes V and directed edges E. The weight matrix $A = (A_{i,j})_{i,j \in V}$ assigns a weight $A_{i,j}$ to every edge $e = e_{i,j}$ in E from node i to j. We use the convention that $A_{i,j} = 0$ if and only if there is no edge from node i to j.

For a node j, we define $S_{\text{in}}(j) \subseteq V$ to be the set of all nodes i in V for which there is an incoming edge from i to j (i.e. $A_{i,j} \neq 0$). Furthermore, $S_{\text{out}}(j) \subseteq V$ are all nodes i in V for which there is an outgoing edge from j to i (i.e. $A_{j,i} \neq 0$).

Finally, every j in V has a node attribute value $0.0 \leq r_j \leq 1.0$. These attributes can initially be determined or can be the result of the risk propagation algorithm performed on a different network with the same nodes.

2.3 Updating attributes

The purpose of the propagation algorithm is to update the attribute values of nodes, using the attributes of neighbouring nodes. This way, the attributes are propagated through the network.

We update the initial attribute value r_j^0 of every node j using the attribute values of the nodes in $S_{\text{in}}(j)$ and the weights in A. A factor δ^k determines the pace of the propagation. For every iteration $k \in \{1, .., K\}$, where K resembles the number of iterations, for every $j \in V$ we update r_j^k by

$$r_j^k = (1 - \delta^k) r_j^{k-1} + \frac{\delta^k}{T_j} \sum_{i \in S_{\text{in}}(j)} r_i^{k-1} A_{i,j}. \tag{1}$$

We define $T_j := \sum_{i \in S_{\text{in}}(j)} A_{i,j}$ as the total incoming weight in node j, the factor $\delta^k := \frac{a}{k^b}$ depends on a fixed $a, b \in \mathbb{R}$ such that with $0 \leq a \leq 1$ and $b \geq 0$ and iteration $k \geq 1$. Note that $0 \leq \delta^k \leq 1$.

The fixed parameters a and b should be determined by the context. The intuition behind the parameters is as follows. The value a decides how significant the attribute of a node is influenced by its neighbors. The value b decides how large δ^k remains over the iterations, and thus how many hops back in the network should have a (significant) influence on determining the final attribute of a node. Now, the higher a is, the more influence neighbouring nodes have on the attribute of an accounts. When $a = 0$, the attributes are completely static while with $a = 1$ (and $b = 0$), the old attribute of a node itself is completely ignored. On the other hand, when performing multiple iterations, an increased value of b leads to an earlier convergence of δ^k and therefore to an earlier stabilization of the attributes of the nodes.

3 Secure Multi-Party Variant

In this section, we present our solution for securely performing the risk propagation algorithm with multiple parties in a privacy-preserving manner. We assume

that the set of nodes V is distributed over multiple parties. Every party knows the attribute values of its own nodes and all (directed) incoming- and outgoing edges associated with its own nodes.

To understand why the algorithm is suitable to be run efficiently in a secure multi-party variant, we note two things. Firstly, if a party wants to update the attribute value of one of its own nodes, it needs the attribute values of the nodes that are neighbouring by incoming edges. In case these neighbouring nodes are owned by a different party, the corresponding attribute values can be received securely in homomorphic encrypted form and used for calculations without decrypting the value. This means that not the entire graph data, but only the attributes of the neighbouring nodes need to be (securely) communicated between the parties. Secondly, since updating an attribute value is a linear operation, this allows for an efficient secure implementation.

3.1 Security model

The secure risk propagation algorithm works in the *passive* security model up to all-but-one corrupted parties (depending on the chosen decryption threshold). This means that, under the assumption that no party actively deviates from the cryptographic protocol, no information can be retrieved about the secret input of an honest party, other than what can be deduced from the outcome of the protocol. The passive security model is especially useful to obtain a relative efficient solution in this collaborative AML setting where the parties do trust each other but are simply not allowed to share private information due to regulations. Note that the security model of an MPC protocol guarantees that no information is leaked to the parties during the computation *except* from what each party could have derived from its own input and output in the protocol. Depending on the computation, some sensitive information could thus still be deduced in practice. We will analyze this further in Section 4.1.

Additive homomorphic encryption We denote encryption $\text{Enc}_{pk}(m)$ for a message m encrypted with public key pk, and we denote $\text{Dec}_{sk}(m)$ for decrypting the encrypted message with the secret key sk. Then by definition $\text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m$. The secure version of the risk propagation algorithm requires the additive homomorphic encryption property, meaning that we can perform addition (but no multiplication) between encrypted values. Furthermore, these schemes allow for the multiplication of a plaintext scalar with an encrypted value. Because of the linearity of risk propagation, there is no need for a less efficient *fully* homomorphic encryption scheme. For the implementation we chose the Paillier encryption scheme [23].

Distributed Key Generation In order to use the Paillier encryption securely in this setting, we need to prevent parties from decrypting intermediate values. Therefore we use an implementation of the encryption scheme with threshold decryption without a trusted dealer, the distributed key generation as described

in [30]. With this encryption scheme, a key pair $\{pk, sk\}$ is generated, where the key pk is public, and the secret key sk is distributed among the parties. In order to decrypt an encrypted value, a certain number of parties is required (the threshold), with each party performing a partial decryption. The decrypted value is only revealed to the parties with access to sufficient partial decryptions. In other words, a ciphertext can only be decrypted if enough parties cooperate. After the distributed key generation, the attribute values are encrypted and the secure In the next sections, we explain how the encrypted values are used to perform secure computations.

Distributed graph The real-world problem of the distributed graph is a transaction graph that is distributed amongst multiple banks. Every bank knows its own accounts including attribute values, and knows the incoming and outgoing transactions, even if they are from or to accounts of another bank. A bank knows which other bank these neighbouring accounts belong to, but does not know their risk scores, and therefore cannot perform the complete risk propagation algorithm.

More mathematically, the input data is a graph as defined in Section 2.2, but now we assume that the graph is distributed amongst t parties, where the set of parties is denoted as $\mathcal{P}$. Every party $P \in \mathcal{P}$ has a unique subset of nodes $V^P \subset V$, such that $V = \bigcup_{P \in \mathcal{P}} V^P$ and for $P, P' \in \mathcal{P}$

$$V^P \cap V^{P'} = \emptyset \text{ if } P \neq P'.$$

We define A^P of A to be all amounts $A_{i,j}$ such that $i \in V^P$ or $j \in V^P$. Also, party P knows the initial attribute values r_j^0 for every $j \in V^P$. Finally, party P knows to which party each of its neighbouring nodes belong. To be more precise, for every $j \in V^P$, for all $i \in S_{\text{in}}(j) \cup S_{\text{out}}(j)$, party P knows P' such that $i \in V^{P'}$.

3.2 Exchange of encrypted attributes

In the distributed graph setting, if party P wants to update the attribute values r_j^0 of $j \in V^P$, according to (1) it needs all r_i^0 for all $i \in S_{\text{in}}(j)$. However, in general not all $i \in S_{\text{in}}(j)$ are in V^P.

Therefore, the first step of secure risk propagation is the parties exchanging the initial attributes they need in an encrypted form. In general, we denote s_i^k as the encrypted form of r_i^k.

If for $i \in S_{\text{in}}(j)$, $i \notin V^P$, party P needs to receive s_i^0 from the party P' such that $i \in V^{P'}$. In order to receive the right encrypted values, the following needs to happen. For all $j \in V^P$, if for $i \in S_{\text{out}}(j)$, there is a party $P' \neq P$ such that $i \in V^{P'}$, then P needs to communicate s_j^0 to P'.

Furthermore, before every new iteration, there is an exchange of the encrypted updated risk scores. See also the *exchange* pseudocode in Algorithm 1. We now explain how the encrypted values are used to update the attributes securely.

Algorithm 1 `exchange`(k, P)

Input: k, s_j^k for all $j \in V^P$, V, A^P
1: **for** $j \in V^P$ **do**
2: **for** $P' \in \mathcal{P} \setminus P$ **do**
3: **if** $\exists i \in S_{\text{out}}(j)$ such that $i \in V^{P'}$ **then**
4: send s_j^k to P'

3.3 Secure Computations

In order to perform the secure computations, every party homomorphically encrypts its attribute values, which we refer to as the attribute ciphertexts. Note that the Paillier scheme uses integers as input, however, the attribute values r_j^k and δ^k are not an integer. For this we use scaling factors, such as in [25]. Since this is a straightforward adaptation, we will ommit the scaling factors in the equations below. In our algorithm, we measured a difference smaller than 10^{-5} between the results of the plaintext- and the secure variant with the scaling factors, which we deem insignificant.

Secure updating of attributes The attribute ciphertexts are exchanged as described in Section 3.2. Using the properties of additive homomorphic encryption, every party computes the updated attribute ciphertexts with an encrypted version of Equation (1). The difference with Equation (1) is that we use homomorphically encrypted ciphertexts. Therefore, sums become products and products with plaintext values become exponentiations as described in Section 3.1. Denoting $s_j^k := \text{Enc}_{pk}(r_j^k)$, we get the following secure update,

$$s_j^k = (s_j^{k-1})^{1-\delta^k} \left[\prod_{i \in S_{\text{in}}(j)} (s_i^{k-1})^{A_{i,j}} \right]^{\frac{\delta^k}{T_j}}. \tag{2}$$

After the first iteration, every party ends up with a ciphertext of the updated attribute values of its own accounts. This value is then exhanged for the next iteration. Before exchanging, every ciphertext is rerandomized by multiplying the ciphertext with a fresh encryption of 0. After rerandomization, the new ciphertexts are exchanged, and another iteration is performed. The pseudocode is found in Algorithm 2.

3.4 Decryption

After performing all iterations, the outcome of the protocol is decrypted with threshold decryption as described in Section 3.1. Every party P sends the ciphertexts of the end results to the other parties, so that they can partially decrypt those values. When party P has received the partially decrypted outcomes, it

Algorithm 2 Secure Risk Propagation algorithm for party P

Input: Public pk, K, δ^k, $\mathcal{P}$
Input: Private V^P, A^P, $(r_j^0)_{j \in V^P}$
Output: Encrypted updated scaled attributes $(s_j^K)_{j \in V^P}$

for $j \in V^P$ **do**
  $s_j^0 \leftarrow \text{Enc}_{pk}(r_j^0)$
$exchange(0, P)$

for $k = 0$ to K **do**
  for $j \in V^P$ upon receiving all $(s_i^k)_{i \in S_{\text{in}}(j)}$ **do**
    $s_j^{k+1} \leftarrow (s_j^k)^{1-\delta^k} \left[\prod_{i \in S_{\text{in}}(j)} (s_i^k)^{A_{i,j}}\right]^{\frac{\delta^k}{T_j}}$
  rerandomize & `exchange`$(k+1, P)$

decrypts the end results with their part of the decryption key. This happens in such a way that only party P learns the updated attribute values of $j \in V^P$.

Also, there are different options for revealing the output of the secure risk propagation. Instead of decrypting all final updated attributes, the parties could decide to reveal only some information about the attribute values. With a *secure comparison*, the ciphertext is transformed into the outcome of a comparison with some threshold. After decryption, only the outcome of this comparison is revealed [29]. Using this technique, the parties could for example decide to only decrypt whether the scores are low, medium or high, or compare it to a certain threshold. This way, only relevant information is revealed, while other information remains private.

4 Results

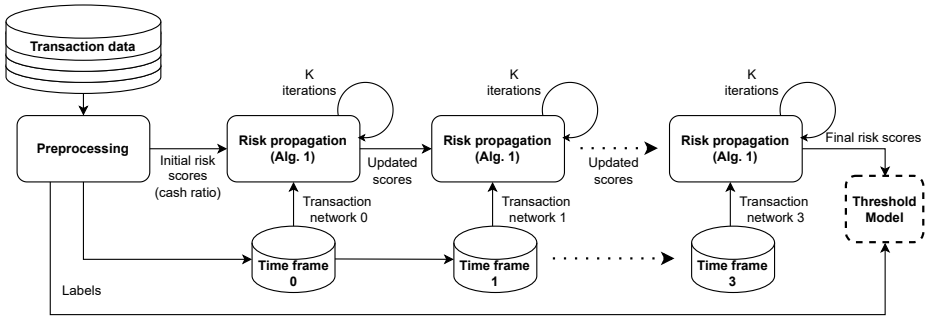
In this section, we first analyze the security of our solution. Afterwards, to measure the performance of the secure risk propagation algorithm, we have performed two types of experiments. First, the predictive power of the algorithm was measured on a real transaction dataset from a large Dutch bank, by using the risk scores computed by our algorithm to predict which accounts should be labeled as having unusual behaviour that requires further investigation. The results of these experiments can be found in Section 4.2.

Secondly, the scalability of the solution was measured by benchmarking the runtime of the algorithm with sufficiently large cryptographic keys to secure the data in practice. The results of this experiment can be found in Section 4.3.

4.1 Security Analysis

As long as the parties follow the algorithm honestly, the only communication that takes place between two banks A and B are encrypted risk scores of accounts from A that make a transaction to an account at B (or vice versa). Since

Fig. 1: Schematic Overview of the Experiment.



the parties both already have this transaction in their respective datasets, this does not leak any additional information about the structure of the underlying transaction network. Furthermore, both parties also already know the amount of the transaction and thus have all the information required to perform the secure update of the risk score. Given that the Paillier encryption scheme is secure, our protocol is thus secure in the passive model.

However, it should be noted that the risk propagation algorithm consists of only linear operations. Therefore, it might be possible to deduce information about the encrypted inputs from only the decrypted results in some cases as explained in Section 3.1. When using this algorithm in practice, this should be prevented. Using secure comparison, the outcome of the secure algorithm can be a category (e.g. low, medium or high) instead of an exact risk score. Also, using multiple iterations where the intermediate outcomes are not decrypted, the output is an outcome of non-linear operations. Lastly, differential privacy [20] could be used to obscure the output of the secure risk propagation, without too much compromise on the performance. The necessity of these measures should be tested in further research.

4.2 Predicting Future Alerts of Unusual Behaviour

A schematic overview of the experiment to measure the predictive power of risk propagation can be found in Figure 1. This experiment was conducted on real transaction data of a single, large bank in the Netherlands. Therefore, we could locally use the plaintext version of the risk propagation algorithm. As explained in Section 3.3, an encrypted version does not significantly influence the results of the algorithm.

Data Selection We filtered the dataset to remove accounts and transactions of minors, deceased persons, transactions via suspense accounts and correspondent banking transactions. Furthermore, we only consider transactions large than 100

euros and aggregate transactions between pairs of accounts. In total, this yields a dataset with a few million bank accounts with an average out degree between 2 and 3. This is data that the bank already uses for their legal obligation under the Dutch WWFT and has not specifically been gathered for this research.

As attributes, the IBAN numbers of the accounts are required as well as the alerts of unusual behaviour that we use to label the dataset. For the transactions, the IBAN of the sender and receiver are required, the transaction amount and an indicator of whether it is a cash transaction or not.

Label information comes from a separate dataset containing, on a *customer level* flags of suspicious behaviour by current AML systems in place. We therefore assign the customer label to all of their accounts. Note that customers being labeled as having unusual behaviour that might be related to money laundering can be labeled for a range of different reasons which does not necessarily have to be related to cash. In the preprocessing step data is extracted for three different periods of time in the data.

Initial attributes. To compute the initial attributes, we chose eight consecutive weeks from September 1, 2021 until October 27, 2021. We use the *cash ratio* as initial attribute. This is computed as the amount of money that was marked as cash that an account has received in this time period, divided by the total amount received. Furthermore, the accounts that made or received at least one transaction in this time period will be the scope of the rest of the experiment.

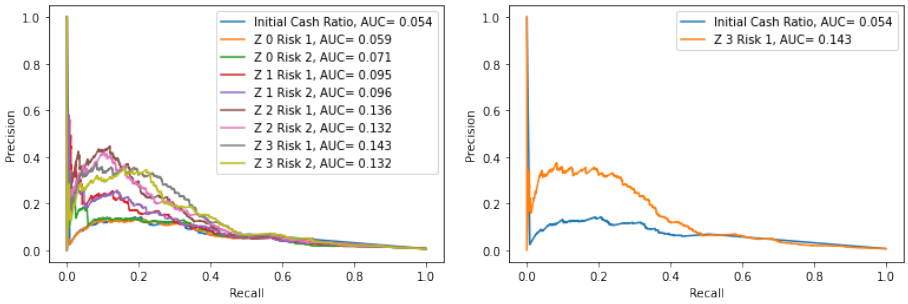
The choice of cash ratio as an informative feature is supported by the three phases of money laundering as described in Section 1. During the placement phase, cash from illicit activities could be brought into the system [14]. We expect that risk propagation algorithm therefore mimics the layering phase.

Transaction network(s). For creating the static transaction network, we use four weeks between September 29, 2021 and October 27, 2021, which we split up into four sub periods of one week each. Note that here, we only build a transaction network for the accounts that are in scope as described above (i.e. that have a cash ratio). The transactions made in a sub period we call a *time frame*. In the remainder of this section, we refer "Z_i" for the i-th time frame. In the experiment, resulting attributes after risk propagation over Z_1, are used as the input attributes for the propagation over Z_2, etcetera. This way we mimic a real time use of risk propagation.

Labels. For assigning labels, we use eight weeks from October 1, 2021 until November 30, 2021. Here we select the alerts that have been raised in this time period. These alerts are used to label the accounts in scope as positive or negative. We deliberately pick an additional month after the transaction networks, since the systems that generate these alerts can have a delay and thus unusual behaviour in October might only be alerted in November.

Choice of parameters As explained in section 2.3, the most suitable values for a and b in the parameter δ^k depend on the context. Using a grid search, we

Fig. 2: PR curves after every iteration. Here, each line is the performance of the model using the resulting risk scores after a certain iteration.



(a) All Iterations. (b) Best- and Initial Model.

determined the choices for our dataset that resulted in (on average) the best models, which were $a = 0.3$ and $b = 0.8$. Furthermore, using a similar strategy we found two iterations per time frame (eight iterations in total) to yield the best results.

Results To measure the performance of the algorithm on the dataset which is highly skewed ($> 99\%$ nodes are negative), we compute precision-recall curves of a threshold model as depicted in Figure 1 after every iteration. The result of the algorithm is a risk score for every account. The model uses these risk scores to predict which accounts are positive by choosing a threshold and mark accounts with a risk score higher than the threshold as positive and negative otherwise. The ground truth is given by the alerts dataset as explained above.

We present the "Area Under the Curve (AUC)" of each model as an indication of how well the models perform. The AUC is calculated as the average precision at each threshold (and thus each recall). Precision describes how likely an account flagged by the risk propagation algorithm is to actually be positive while recall describes how well the algorithm performs at finding the positive accounts. After each iteration, we evaluate the predictive power of the risk scores by computing the precision and recall for a range of thresholds. Intuitively, a lower threshold means we have a higher recall since we flag more accounts as positive at the cost of a lower precision.

The results of this experiment can be found in Figure 2a. The AUC of simply guessing which accounts are positive would precisely equal the percentage of positive accounts, which is less than 1%. Now, the initial cash ratio shows how well a model just using the cash ratio as an indicator for a positive account would perform. As expected, there is some correlation between higher cash ratios and accounts being labeled as positive, which results in an AUC that is already better than randomly guessing.

As can be seen, the AUC further improves as we perform more iterations of the algorithm. The best results for this dataset and with these parameters are after seven iterations, which is the first iteration of the last time frame (Z_3) . With an AUC of 0.143, we see that this model performs almost three times better than the initial cash ratio model. This gives a strong indication that propagating risk scores through transaction networks supports finding more accounts with unusual behaviour that might be harder to find when looking at each account individually.

In Figure 2b, only this best model and the initial model are presented to give a better view on how the model improved. With the optimal threshold choice of this model, it reaches a precision of 40% and a recall of 25%. This means the model is able to find 25% of all positive accounts and from all the accounts that it did mark as positive, 40% were correctly labeled as positive. On the other hand, the model using just the initial cash ratios has a precision of 15% for the same recall. Therefore propagating is useful to drastically reduce the amount of false positives of such an algorithm.

In general, we see that the models improve the most after the first iteration in a new transaction network time frame. This is likely due to the fact that in those iterations, new information is introduced (namely, new transactions).

4.3 Scalability of Secure Variant

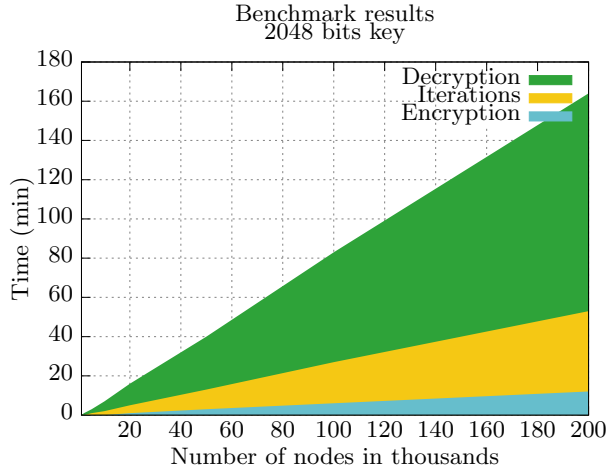
In order to test scalability, we benchmarked the secure variant of the risk propagation algorithm. The secure implementation of our risk propagation is open source [16]. The implementation is made in Python and uses the Gmpy2 library to perform faster modular arithmetic. The distributed Paillier key used during the experiments is 2048 bits in size. Each party runs in its own process. Each process has access to a single core of 3.5GHz in a cloud environment. The processes communicated with each other via HTTP connections. For the benchmarking experiments random graphs of up to 200,000 accounts were generated, with the accounts equally divided over three parties. Then for each account, incoming transactions were sampled according to a Poisson distribution with a mean of 10. The corresponding transaction amounts were chosen between 0 and 1000 while risk scores for all the accounts were sampled ranging from 0 to 1.

As for the choice of parameters, the choices for δ^k is irrelevant for the performance of the secure risk propagation. We performed $k = 2$ iterations. This represents processing one transaction network in the experiments in 4.2.

Results In Figure 3 the execution time is split into three sections. Firstly the encryption of the cash ratios of all the accounts, then the iterations to propagate and update the encrypted risk scores and finally decryption to reveal the resulting risk scores to the respective parties.

In this scenario the entire algorithm for $200,000$ accounts ran within three hours with the majority of the runtime consisting of decrypting the updated attribute values. Note that the decryption time is constant for the same number

Fig. 3: Scalability benchmark with distributed Paillier key of 2048 bits.



of nodes. Therefore, for larger numbers of iterations, the decryption time is negligible. Finally, we observe that the runtime of all parts are linear in the number of accounts in the dataset. This gives a strong indication that the algorithm is scalable and feasible for analyzing larger datasets with more banks involved. From the linearity we can predict that the algorithm can be run on millions of bank accounts in a reasonable amount of time (e.g. a few days).

5 Conclusion

In this paper we presented a novel, efficient algorithm for the detection of money laundering using the graph structure of a transaction network. Our results on real bank data strongly indicate that the propagated risk scores can improve the detection of bank accounts involved with unusual behaviour that requires further investigation.

Furthermore, due to the conceptual simplicity of the algorithm, we have shown that it can easily be extended to a secure multi-party version that is feasible in practice. The secure version of the algorithm can be used to make the risk scores more informative by combining data from multiple banks. This gives a group of banks a way to cooperate and share insight with each other using information which they are otherwise not allowed to share.

Finally, looking at the performance of our algorithm, we believe risk propagation should not be treated as a stand-alone solution for doing AML. Rather, it should be used as an additional feature to further strengthen the knowledge of existing detection algorithms by bringing in additional information from the transaction graph. This is supported by the observation that risk propagation is particularly good at reducing false positives. Since each account flagged as

positive needs to be manually investigated by an analyst, this is believed to significantly strengthen the intelligence position of banks.

Discussion & Future Work Firstly, the experiments on real bank data in Section 4.2 show that the risk propagation has predictive value. However, the predictive value on a network of multiple banks has not yet been tested. Intuitively, we expect this to be even better, since performing the algorithm in a multi-party setting provides an improved view on the entire network and money laundering patterns take place over multiple banks. This should be tested in a pilot setting. Furthermore, the optimal choice for the time periods should be found experimentally in practice. Our hypotheses is that the results improve with smaller time periods, ultimately reaching real time transaction monitoring. Of course this would require more computational resources or efficiency improvements in the algorithm. Additionally, we have only measured the predictive power aggregated over all the accounts but could not verify whether the algorithm indeed detects new money laundering patterns.

Secondly in a multi-bank setting, it is important that the banks agree on the semantics of the attributes. If banks use a different definition for the attributes, the multi-party risk propagation will be less useful. However, from competition law, it is advisable to carefully consider which attributes to use and exchange such that this is lawful. The banks need to agree on the used attribute such that it is based on an objective indicator and not an indicator that is subject to interpretation of how a bank perceives risks. A good example of such an objective indicator is the cash ratio, which is well-defined and objectively measured without any interpretation of risk involved. How each bank uses the updated risk scores is up to each bank individually and this is not shared between the banks.

Thirdly, the alerts used to label our dataset for the accuracy experiments are on a customer level while the risk propagation was done over bank accounts. However in practice, some of these accounts might not have anything to do with this behaviour and are thus impossible to find, resulting in additional false negatives. Furthermore, the alerts dataset contains alerts for any unusual activity in the light of the Dutch WWFT (law for preventing money laundering and terrorism financing). These might not have anything to do with cash while cash is the only feature we used for risk propagation. This might have impacted our results. Therefore, our results should be seen as a "baseline" for the predictive power of the risk propagation algorithm but it is hard to determine its performance in practice.

Finally, our secure solution is secure in the passive security model. This is sufficient for parties who do trust each other. In case this assumption is not realistic, the stronger model of active security should be adopted, which typically introduces additional overhead. It would therefore be interesting to research how this can be done in an efficient manner for the risk propagation algorithm.

A Notation

The following notation is used in Algorithm 1 and Algorithm 2.

pk	Paillier public key
$\text{Enc}_{pk}(m)$	Encryption of message m using public key pk
K	number of iterations
δ^k	multiplication factor $\frac{a}{k^b}$ depending on k and pre-defined a and b
$\mathcal{P}$	Set of parties
V	Set of all nodes in transaction network
V^P	Set of all nodes belonging to party P
A	Weight matrix of transaction network
A^P	All in- and outgoing weights of party P
T_j	Total incoming weight for node j
$S_{\text{in}}(j)$	All nodes i such that $A_{i,j} \neq 0$
$S_{\text{out}}(j)$	All nodes i such that $A_{j,i} \neq 0$
r_j^0	Initial attribute of node j
r_j^k	Attribute of node j after k iterations
s_j^k	Encrypted attribute of node j after k iterations

Table 1: Overview of notation.

References

1. Mousa Albashrawi. Detecting financial fraud using data mining techniques: A decade review from 2004 to 2015. *Journal of Data Science*, 14(3):553–569, 2016.
2. Alhanouf Abdulrahman Saleh Alsuwailem and Abdul Khader Jilani Saudagar. Anti-money laundering systems: A systematic literature review. *Journal of Money Laundering Control*, 23(4):833–848, 2020.
3. Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Van Vyve. Securely solving simple combinatorial graph problems. In *International Conference on Financial Cryptography and Data Security*, pages 239–257. Springer, 2013.
4. Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. Secure graph analysis at scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 610–629, 2021.
5. Carsten Baum, James Hsin-yu Chiang, Bernardo David, and Tore Kasper Frederiksen. Sok: Privacy-enhancing technologies in finance. *Cryptology ePrint Archive*, 2023.
6. Mark Button, Branislav Hock, and David Shepherd. *Economic Crime: From Conception to Response*. Routledge, United Kingdom, 1st edition, April 2022.
7. Andrea Fronzetti Colladon and Elisa Remondi. Using social network analysis to prevent money laundering. *Expert Systems with Applications*, 67:49–58, 2017.

8. Daniele Cozzo and Nigel P Smart. Secure fast evaluation of iterative methods: With an application to secure pagerank. In *Topics in Cryptology–CT-RSA 2021: Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17–20, 2021, Proceedings*, page 1. Springer Nature, 2021.
9. John Cusack. Global threat assessment. Technical report, Financial Crime News, 2019.
10. Paola de Perthuis and David Pointcheval. Two-client inner-product functional encryption with an application to money-laundering detection. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 725–737, 2022.
11. Rafał Dreżewski, Jan Sepielak, and Wojciech Filipkowski. The application of social network analysis algorithms in a system supporting money laundering detection. *Information Sciences*, 295:18–32, 2015.
12. EUR-lex. Access to European Union Law. `https://eur-lex.europa.eu/EN/legal-content/glossary/money-laundering.html`. [Online; accessed 17-May-2023].
13. Jingguang Han, Yuyun Huang, Sha Liu, and Kieran Towey. Artificial intelligence for anti-money laundering: a review and extension. *Digital Finance*, 2(3):211–239, 2020.
14. Angela Samantha Maitland Irwin, Kim-Kwang Raymond Choo, and Lin Liu. An analysis of money laundering and terrorism financing typologies. *Journal of Money Laundering Control*, 2012.
15. Dimitar Jetchev, N Gama, K Georgieva, J McCarthy, A Odersky, Petric, and A Sae-Tang. Detecting money laundering activities via MPC network flow analysis. `https://www.youtube.com/watch?v=_nXbLGEIi98&t=2553s`, 2020. Real World Crypto.
16. TNO PET Lab. *Secure Risk Propagation*. `https://github.com/TNO-MPC/protocols.risk_propagation`.
17. N Maxwell. Innovation and discussion paper: Case studies of the use of privacy preserving analysis to tackle financial crime. `https://www.future-fis.com/uploads/3/7/9/4/3794525/ffis_innovation_and_discussion_paper_-_case_studies_of_the_use_of_privacy_preserving_analysis_-_v.1.3.pdf`, January 2021.
18. Krzysztof Michalak and Jerzy Korczak. Graph mining approach to suspicious transaction detection. In *2011 Federated conference on computer science and information systems (FedCSIS)*, pages 69–75. IEEE, 2011.
19. Ministerie van Financiën and Ministerie van Justitie en Veiligheid. Algemene leidraad wet ter voorkoming van witwassen en financieren van terrorisme (wwft), 7 2020.
20. Tamara T Mueller, Dmitrii Usynin, Johannes C Paetzold, Daniel Rueckert, and Georgios Kaissis. Sok: Differential privacy on graph-structured data. *arXiv preprint arXiv:2203.09205*, 2022.
21. Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. Graphsc: Parallel secure computation made easy. In *2015 IEEE Symposium on Security and Privacy*, pages 377–394. IEEE, 2015.
22. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
23. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *International conference on the theory and applications of cryptographic techniques*, pages 223–238, 1999.

24. Ahmad Salehi, Mehdi Ghazanfari, and Mohammed Fathian. Data mining techniques for anti money laundering. *International Journal of Applied Engineering Research*, 12(20):10084–10094, 2017.
25. Alex Sangers, Maran van Heesch, Thomas Attema, Thijs Veugen, Mark Wiggerman, Jan Veldsink, Oscar Bloemen, and Daniël Worm. Secure multiparty pagerank algorithm for collaborative fraud detection. In *International Conference on Financial Cryptography and Data Security*, pages 605–623. Springer, 2019.
26. Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Alaa Yagoub, and Aijun An. A new algorithm for money laundering detection based on structural similarity. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 1–7. IEEE, 2016.
27. Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, Guangnan Ye, Keith Houck, Ryo Kawahara, Ali Anwar, Lucia Larise Stavarache, Yuji Watanabe, Pablo Loyola, et al. Towards federated graph learning for collaborative financial crimes detection. *arXiv preprint arXiv:1909.12946*, 2019.
28. Marie Beth van Egmond, Thomas Rooijakkers, and Alex Sangers. Privacy-preserving collaborative money laundering detection. *ERCIM NEWS*, page 27, 2021.
29. Thijs Veugen. Encrypted integer division and secure comparison. *International Journal of Applied Cryptography*, 3(2):166–180, 2014.
30. Thijs Veugen, Thomas Attema, and Gabriele Spini. An implementation of the paillier crypto system with threshold decryption without a trusted dealer. *IACR Cryptol. ePrint Arch.*, 2019:1136, 2019.
31. Arman Zand, James Orwell, and Eckhard Pfluegel. A secure framework for anti-money-laundering using machine learning and secret sharing. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–7. IEEE, 2020.