Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection

Alex Sangers[1(✉)], Maran van Heesch[1], Thomas Attema[1,5], Thijs Veugen[1,5], Mark Wiggerman[2], Jan Veldsink[3], Oscar Bloemen[4], and Daniël Worm[1]

[1] Netherlands Organisation for Applied Scientific Research (TNO), The Hague, The Netherlands
alex.sangers@tno.nl
[2] ABN AMRO, Amsterdam, The Netherlands
[3] Rabobank, Utrecht, The Netherlands
[4] ING, Amsterdam, The Netherlands
[5] CWI, Amsterdam, The Netherlands

Abstract. Collaboration between financial institutions helps to improve detection of fraud. However, exchange of relevant data between these institutions is often not possible due to privacy constraints and data confidentiality. An important example of relevant data for fraud detection is given by a transaction graph, where the nodes represent bank accounts and the links consist of the transactions between these accounts. Previous works show that features derived from such graphs, like PageRank, can be used to improve fraud detection. However, each institution can only see a part of the whole transaction graph, corresponding to the accounts of its own customers. In this research a new method is described, making use of secure multiparty computation (MPC) techniques, allowing multiple parties to jointly compute the PageRank values of their combined transaction graphs securely, while guaranteeing that each party only learns the PageRank values of its own accounts and nothing about the other transaction graphs. In our experiments this method is applied to graphs containing up to tens of thousands of nodes. The execution time scales linearly with the number of nodes, and the method is highly parallelizable. Secure multiparty PageRank is feasible in a realistic setting with millions of nodes per party by extrapolating the results from our experiments.

Keywords: Multiparty computation · PageRank · Fraud detection · Collaborative computation

1 Introduction

Cyber security, anti-fraud and other anti-crime activities benefit from cooperation amongst involved parties like financial institutions, governments and law enforcement agencies. The public and private sectors are actually stimulated by regulators to perform joint activities and share threat intelligence and other

I. Goldberg and T. Moore (Eds.): FC 2019, LNCS 11598, pp. 605–623, 2019.
https://doi.org/10.1007/978-3-030-32101-7_35

data as they have a common goal to battle this type of crime. Examples of such data are lists of known criminals, confirmed money mules and known malicious IP addresses. Sharing benign operational data on customers, transactions and events between different organizations would be beneficial as well. However, sharing benign data between organizations has always been strongly limited due to competition and privacy regulations, especially if it concerns personal data of customers and employees. The risks of sharing data for companies as well as public services are loss of trust in services, integrity, financial losses, societal damage and damaged reputation.

1.1 The Financial Sector

The financial sector continuously fights the misuse of the financial infrastructure for criminal activities like fraud and money laundering. An example of such a criminal activity is the following.

Example 1 ('Carousel'). *Loan applications are based on income of the client that requests the loan. A criminal may try to feign income by creating repeated transactions to his account, or node, coming from another node pretending to be a company - just as legitimate salary payments. To keep the needed funds for a criminal low, the money is often drained from the account and placed back on the node of the fake company where the process is repeated.*

By looking at the whole network, we may quickly realize that while the feigned salary payments look similar to other salary payments, the node pretending to be a company lacks the structure we see of nodes known to be companies.

Protected by privacy regulations such as the recent GDPR, these bank-transcending fraud and money laundering cases can be challenging to detect. In fact, even malign transaction sequences moving through different departments or channels within a bank may be troublesome to detect, due to the confidentiality of the involved data.

Financial crime detection is an example of a situation in which different parties share a common interest, but confidentiality and privacy regulations prevent collaboration. In a payment transaction a financial institution typically only knows one of the parties involved in the payment. Financial institutions would greatly benefit from accessing information from other organizations.

1.2 Secure Multiparty Computation

Secure multiparty computation (MPC) provides a cryptographic solution to the described dilemma above. MPC protocols are cryptographic techniques that allow multiple parties to collaboratively evaluate a function on private input data in such a way that only the output of the function is revealed, i.e. private input remains private. MPC could be explained as the implementation of a trusted third party that collects all relevant input data, evaluates the desired function and reveals its output. However, using an actual trusted third party,

such as a consultancy agency, to collect and analyse all private information is often not allowed by regulations and usually expensive.

Already in the 1980s it was shown that any computable function can be evaluated securely, i.e. in an MPC fashion [3,9,14,27,28]. However, early MPC protocols came at a cost as they introduced a significant computation and/or communication overhead. Over the years progress has been made and research interests have shifted towards practical applicability making MPC ready for deployment [4,12,17,19,22].

MPC has been applied to various use cases ranging from sugar-beet auctions [5] to key-management systems [25]. Moreover, applications in the financial domain include confidential benchmarking [10] and off-exchange trading [24]. All these use cases fall under the MPC paradigm in which multiple parties aim to collaboratively evaluate some function without revealing its private input values.

Secure graph algorithmic has been another particular area of interest. Shortest path and max-flow algorithms, for example, find their applicability in many situations and a natural question to ask is whether these algorithms can be evaluated in a privacy-preserving manner. In [8], a secure shortest path algorithm is constructed for the 2-party setting. In [1], the shortest path and max-flow algorithm are considered in the general multi-party setting. However, the complexity of these algorithms renders them only applicable to small graphs.

In 2015, Nayak et al. [21] developed a framework for securely computing graph algorithms like PageRank. The main difference is that they outsource the secure graph algorithm to two parties, who execute a garbled circuit. In our solution, the partial graph owners jointly perform the algorithm by means of additively homomorphic encryption. Their solution has complexity $\mathcal{O}(M \log M)$, where $M = |V| + |E|$, because edges and nodes need to be obliviously sorted. We exploit the fact that in our setting each party knows its own transaction graph, because then additively homomorphic encryption allows for local computations with private values, and sorting is not necessary, which leads to an overall $\mathcal{O}(M)$ complexity. Both solutions can easily be parallellised.

MPC delivers the mechanisms needed to collaborate and safeguard data security and privacy without the need for a trusted third party, which would be highly beneficial for the financial industry.

The rest of this paper is structured as follows. The PageRank algorithm is explained in Sect. 2. Secure multiparty PageRank is described in Sect. 3. In Sect. 4 the performance results are presented and the conclusions are presented in Sect. 5.

2 PageRank for Fraud Detection

In a transaction graph, nodes represent bank accounts, and edges consist of the unique transactions between accounts. Several graph-based features can be used in machine-learning algorithms to improve existing fraud detection algorithms, by reducing the false positives of existing techniques [20]. Namely, after the graph-based features are computed, new transactions that are classified as fraudulent by

an existing fraud-detection technique can be re-evaluated by an algorithm that uses these features.

One of these graph-based features is *PageRank*, developed by Google to return a ranking of websites when searching on the web. In essence, PageRank is a centrality measure for all nodes in the directed graph, and can be used for other purposes as well. Together with other features, PageRank and reverse PageRank (PageRank on the reversed directed graph) have shown their value in discriminating between fraudulent and non-fraudulent transactions [20].

2.1 Requirements for PageRank Application for Fraud Detection

Financial institutions could compute the PageRank values of bank accounts with their observation of the transaction graph. However, in that case they would use only a part of the whole transaction graph. The PageRank values would be much more accurate if the PageRank algorithm were securely applied on transaction data of multiple financial institutions.

In order to apply secure multiparty PageRank in the fraud detection of a bank, the PageRank values need to be available as a feature for machine learning models that use PageRank as input feature. They need to be calculated based on the transaction graph for a predefined period, which is typically one or two months, and need to be updated regularly, e.g. on a monthly basis.

This requires the PageRank and the reverse PageRank computations to take place within $\sim$15 days each. For practical application, however, it would be preferable to compute the PageRank values within $\sim$1 day. This process spans from the starting point where all participants have their graph for the given period ready, to the moment when the PageRank values for all nodes of the participants have been computed and can be used. Furthermore, a reasonable bandwidth for each participant is required, e.g., 100 Mbit/s.

In terms of information, it is not allowed for any participant to learn anything about the graph of another participant, other than what can be learned from the final private PageRank values.

2.2 PageRank and the Power Method

The original goal of the PageRank algorithm is to compute a scalable centrality measure for websites using the hyperlink structure of the web. Intuitively, imagine an Internet user that randomly follows hyperlinks on websites, goes to the next website, etc. As soon as it encounters a website without hyperlinks (a so-called *dangling* website), a new website is chosen at random. In addition to this behavior of following hyperlinks, the Internet user will 'teleport' to any random website with a known probability $1-p$. The resulting probability distribution of visiting frequencies of the Internet user on the websites represents the PageRank value of each website.

Inspired by [20], this idea can easily be translated to transaction graphs. Similarly as with the websites, as soon as a dangling node is encountered, a

new node is chosen at random, and there is a teleport probability $1 - p$. The PageRank values of the transaction graph is the probability distribution based on the visiting frequency of an imaginary coin on the bank accounts.

A commonly used iterative solution method to compute the PageRank is the so-called *power method*. A useful variant thereof has been presented in [18], where only the PageRank values of non-dangling nodes have to be computed during the power method iterations, while correcting for the contribution of dangling nodes. Moreover, this variant has the added benefit of closely matching the intuition given at the beginning of this subsection. The set of dangling nodes is denoted by D and the set of non-dangling nodes is denoted by U. The PageRank value of node j at the k-th iteration is denoted as x_k^j. Equation (1) describes the initialization and iterations of the power method variant of [18] to compute the PageRank values of the non-dangling nodes.

$$\begin{aligned} x_0^j &= \frac{1}{n}, \quad \forall j \in U. \\ x_{k+1}^j &= \frac{1-p}{n} + p \cdot \sum_{i \in S(j)} \frac{x_k^i}{c_i} + \frac{p}{n}(1 - \sum_{i \in U} x_k^i), \quad \forall j \in U, \\ &= \frac{1}{n} + p \cdot \sum_{i \in S(j)} \frac{x_k^i}{c_i} - \frac{p}{n} \sum_{i \in U} x_k^i, \quad \forall j \in U, \end{aligned} \tag{1}$$

where p is a fixed probability, n the total number of nodes, c_i the out-degree of node i, $S(j)$ is the set of incoming edges of node j intersected with U. Note that $c_i \geq 1$ for $\forall i \in S(j)$. Equation (2) describes how the PageRank values of the dangling nodes can be computed after convergence of the power method iterations in Eq. (1).

$$x^j = \frac{1}{n} + p \cdot \sum_{i \in S(j)} \frac{x^i}{c_i} - \frac{p}{n} \sum_{i \in U} x^i, \quad \forall j \in D. \tag{2}$$

Under mild conditions that are satisfied by transaction graphs, the convergence rate of the power method equals p [15] (and is thus independent of the size of the graph). For the commonly used $p = 0.85$, the power method converges within 50 to 100 iterations.

3 A Secure Multiparty PageRank Algorithm

Several factors make securely implementing an algorithm a non-trivial task. The overhead introduced by MPC is significant, requiring a careful analysis of the PageRank algorithm in order to select the optimal MPC protocol. Moreover, most cryptographic protocols work over finite groups, rings or fields and not over the real or complex numbers. This requires a specific representation of the PageRank algorithm, which is originally defined over the real numbers.

3.1 Additively Homomorphic Encryption

A key observation is that the PageRank algorithm consists of mainly linear operations, i.e. additions and multiplications by constants. It is assumed that the total number of nodes n (bank accounts) and the PageRank probability p are publicly known constants, and therefore the only non-linearity in Eq. (1) is division by the private value c_i. The variable c_i represents the number of outgoing edges of node $i \in V$, is fixed throughout the algorithm and is known by the associated party, for which this division can thus be seen as a linear operation.

A crucial observation is that all nodes i are owned by one of the parties participating in the protocol. In practice, this does not have to be the case as there might be transactions to accounts owned by other banks. To take into account these nodes, other approaches, that are out-of-scope for this paper, are required.

An approach to utilize the linear properties of the PageRank algorithm is additively homomorphic encryption. A homomorphic encryption scheme allows the evaluation of certain functions on encrypted input values while remaining oblivious to the actual input values.

Any of the parties could play the evaluator role and perform the computations, as long as the other parties deliver their encrypted input values. The computations can also be distributed amongst the parties so that they share the computational effort. By distributing the computations in such a way that the division by the private value c_i is executed by the party that knows this value, all computations become linear, i.e. ciphertexts do not have to be multiplied by other ciphertexts. Because of this linearity the encryption scheme only has to be additively homomorphic and there is no need to use the more sophisticated but far less efficient fully homomorphic encryption (FHE) schemes. For this reason, the additively homomorphic Damgård-Jurik encryption scheme [11], which is a generalization of the Paillier encryption scheme [23], has been adapted.

Damgård-Jurik is a public-key encryption scheme that takes plaintexts from $\mathbb{Z}_{N^s}$ and maps them to ciphertexts in $\mathbb{Z}^*_{N^{s+1}}$, for some $s \in \mathbb{Z}_{>0}$,

$$\mathrm{Enc}_{pk} : \mathbb{Z}_{N^s} \rightarrow \mathbb{Z}^*_{N^{s+1}},$$

where N is an RSA-modulus and pk is the public key with the associated private key sk. The Damgård-Jurik encryption function is probabilistic; it takes as additional input a random argument $r \in_R \mathbb{Z}_{N^{s+1}} \backslash \{0\}$ for each invocation, which we omit in our notation. The additive homomorphic property means that for all $a, b \in \mathbb{Z}_N$,

$$\mathrm{Dec}_{sk}\left(\mathrm{Enc}_{pk}\left(a\right) \cdot \mathrm{Enc}_{pk}\left(b\right)\right) = \mathrm{Dec}_{sk}\left(\mathrm{Enc}_{pk}\left(a+b\right)\right) = a + b \mod N^s,$$

and, as a consequence, for all $c \in \mathbb{Z}$,

$$\mathrm{Dec}_{sk}\left(\mathrm{Enc}_{pk}\left(a\right)^c\right) = \mathrm{Dec}_{sk}\left(\mathrm{Enc}_{pk}\left(c \cdot a\right)\right) = c \cdot a \mod N^s.$$

The parameter s influences the size of the plaintexts, the size of the ciphertexts and the ratio of the former two. In Sect. 3.6 we will see that, in our case, $s = 1$

results in a sufficiently large plaintext space. For this reason we will fix $s = 1$ from now on.

It must also be noted that this approach of distributing the computations does introduce a communication overhead in contrast to using, for example, a fully homomorphic encryption scheme. This trade-off between computation and communication complexity is typical for applying MPC. As we will see later in Sect. 4, the communication overhead of our solution is acceptable.

3.2 PageRank Algorithm over $\mathbb{Z}_N$

The PageRank algorithm is defined over the real numbers, whereas the Damgård-Jurik encryption scheme assumes plaintexts in the finite ring $\mathbb{Z}_N$. To solve this discordance the PageRank algorithm has to be defined over $\mathbb{Z}_N$ such that the outcome (approximately) coincides with the outcome of the original PageRank algorithm.

An integer representative y of the real number x can be found by applying a scaling factor $f_x \in \mathbb{Z}_+$,

$$y = \lfloor f_x \cdot x \rceil \in \mathbb{Z}.$$

The fixed scaling factor f_x determines the precision of the computations, in fact, the real number x can be approximated by $\frac{y}{f_x}$ and

$$\left| x - \frac{y}{f_x} \right| \leq \frac{1}{2f_x}.$$

The scaling factor f_c is applied to find an integer representation of the fraction $\frac{p}{c_i}$. Multiplying both sides of Eq. (1) with the factor $(f_c)^{k+1} f_x$ then results in the following recurrence relation:

$$(f_c)^{k+1} f_x x_{k+1}^j = \frac{(f_c)^{k+1} f_x}{n} + \sum_{i \in S(j)} \frac{p f_c}{c_i} (f_c)^k f_x x_k^i - \frac{p f_c}{n} \sum_{i \in U} (f_c)^k f_x x_k^i.$$

Defining $\tilde{y}_k^j := \left\lfloor (f_c)^k f_x x_k^j \right\rceil$, $\phi_k := \left\lfloor \frac{(f_c)^{k+1} f_x}{n} \right\rceil$, $\rho_i := \left\lfloor \frac{p f_c}{c_i} \right\rceil$ and $\psi := \left\lfloor \frac{p f_c}{n} \right\rceil$ for all nodes i and iterations k yields the following approximation:

$$\tilde{y}_{k+1}^j \approx \phi_k + \sum_{i \in S(j)} \rho_i \tilde{y}_k^i - \psi \sum_{i \in U} \tilde{y}_k^i.$$

From this the recurrence relation of Eq. (3) is deduced, which is defined over the integers and can be used to approximate the PageRank values of the non-dangling nodes.

$$y_{k+1}^j = \phi_k + \sum_{i \in S(j)} \rho_i y_k^i - \psi \sum_{i \in U} y_k^i \quad \text{with} \quad y_0^j = \left\lfloor f_x x_0^j \right\rceil, \quad \forall j \in V. \tag{3}$$

The same scaling approach is applied to Eq. (2) in which the PageRank values for the dangling nodes are computed. The PageRank values x_k^i can be approximated by $y_k^i/((f_c)^k f_x)$ with a precision that can be modified by changing the scaling factors f_x and f_c.

It is now straightforward to define the PageRank algorithm over $\mathbb{Z}_N$, where the RSA-modulus N should be chosen such that we avoid modular reductions (or overflows) during the evaluation of the PageRank algorithm. It is easy to see that more precision, or larger scaling factors, requires N to be larger as well. In Sect. 3.6 the exact parameter choices will be presented.

3.3 PageRank Algorithm in the Encrypted Domain

The encryption function Enc_{pk} maps Eq. (3) to the following recursive relation over $\mathbb{Z}_{N^2}^*$,

$$z_{k+1}^j = \Phi_k \cdot \prod_{i \in S(j)} \left(z_k^i\right)^{\rho_i} \cdot \left(\prod_{i \in U} z_k^i\right)^{-\psi} \mod N^2 \quad \forall k,$$
$$z_0^j = \mathrm{Enc}_{pk}\left(y_0^j\right) \quad \forall j \in V.$$

Here $\rho_i, \psi \in \mathbb{Z}$ are as in Sect. 3.2, $\Phi_k := \mathrm{Enc}_{pk}(\phi_k) \in \mathbb{Z}_{N^2}^*$ and $z_k^j \in \mathbb{Z}_{N^2}^*$ for all j, k. Since the encryption scheme is additively homomorphic it follows that $\mathrm{Dec}_{sk}\left(z_k^j\right) = y_k^j$ for all j, k.

Note that the value ρ_i is derived from the out-degree of node $i \in V$, and therefore contains private information. For this reason, the terms $\left(z_k^i\right)^{\rho_i}$ should be computed by the party owning node i. Similarly, the sets $S(j)$ and U contain private information, hence summing over these sets can only be done collaboratively. More precisely, the product $\prod_{i \in S(j)} \left(z_k^i\right)^{\rho_i}$ should be computed by the party owning node j, and the product $\prod_{i \in U} z_k^i$ should be computed as the product of all privately computed products $\prod_{i \in U^P} z_k^i$, for all parties $P \in \mathcal{P}$. Here U^P is the set of non-dangling nodes belonging to party P and $\mathcal{P}$ is the set of all parties.

Algorithm 1 describes the computations that have to be performed by party P, assuming that the encryption key has already been generated. Except for the initialization phase all computations take place in $\mathbb{Z}_{N^2}^*$, hence modular reductions are implicit. Each party executes this algorithm and it is evident that it can only be performed collaboratively.

All values that are broadcast are ciphertexts, hence they do not leak private information. However, together multiple ciphertexts might leak information. In particular, we see that ρ_j could be derived from the two ciphertexts Z_k^j and z_k^j. For this reason, every ciphertext is rerandomized before it is broadcast. By rerandomization we obtain another unlinkable ciphertext that decrypts to the same value. We therefore maintain the required functionality without leaking private information. Rerandomization is a standard technique and, in our case, comes down to multiplying the ciphertext with a fresh encryption of 0.

Furthermore, the broadcasting introduces unnecessary communication and even leaks some private information, namely the set U^P. Both can be avoided by only sending the ciphertexts to the parties that require them.

Algorithm 1. Secure PageRank algorithm for party P

Public inputs: pk, n, p, f_x, f_c
Private input: $U^P, D^P, (c_j)_{j \in V^P}, (S(j))_{j \in V^P}$
Output: Encrypted and scaled PageRank values $\left(z_K^j\right)_{j \in V^P}$

$\psi \leftarrow \left\lfloor \frac{p f_c}{n} \right\rceil$ ▷ Initialization
for $j \in U^P$ **do**
  $z_0^j \leftarrow \mathrm{Enc}_{pk}\left(\left\lfloor \frac{f_x}{n} \right\rceil\right)$
  $\rho_j \leftarrow \left\lfloor \frac{p f_c}{c_j} \right\rceil$
for $k = 0$ to $K - 1$ **do** ▷ Non-dangling nodes
  $\Phi_k \leftarrow \mathrm{Enc}_{pk}\left(\left\lfloor \frac{(f_c)^{k+1} f_x}{n} \right\rceil\right)$
  $s_k^P \leftarrow \left(\prod_{i \in U^P} z_k^i\right)^{-\psi}$
  Rerandomize and broadcast s_k^P
  Upon receiving s_k^Q for all $Q \in \mathcal{P}$ **do**
    $S_k \leftarrow \prod_{Q \in \mathcal{P}} s_k^Q$
  for $j \in U^P$ **do**
    $Z_k^j \leftarrow \left(z_k^j\right)^{\rho_j}$
    Rerandomize and broadcast $\left(j, Z_k^j\right)$
  for $j \in U^P$ **do**
    Upon receiving Z_k^i for all $i \in S(j)$ **do**
      $z_{k+1}^j \leftarrow \Phi_k \cdot S_k \cdot \prod_{i \in S(j)} Z_k^i$
for $j \in D^P$ **do** ▷ Dangling nodes
  $z_K^j \leftarrow \Phi_{K-1} \cdot S_{K-1} \cdot \prod_{i \in S(j)} Z_{K-1}^i$

3.4 Key Generation and Decryption

In conventional deployments of public key cryptosystems (e.g. securing communication channels) both the public pk and private key sk are generated by one of the parties. In our setting, however, giving one of the parties complete knowledge of the private key would undermine our privacy requirements. For this reason, our solution requires a distributed implementation of the key generation and decryption algorithm, ensuring that ciphertexts can only be decrypted collaboratively, while individually no information can be deduced. Solutions for distributed key-generation and decryption, allowing up to $|\mathcal{P}| - 1$ passive corruptions, are readily available for some cryptosystems [11,16], including, in particular, the DJ scheme of our choice.

3.5 Security Model

The implementation achieves computational security in the semi-honest model (passive security), i.e. assuming that all parties follow the prescribed protocol. The number of passive corruptions that can be tolerated is $|\mathcal{P}| - 1$.

It is assumed that the number of nodes n in the entire graph is publicly known. This can be achieved by each party P sharing their number of nodes n^P, or by again applying an MPC solution to avoid leaking n^P, and only revealing n. In our solution the values are n^P are made public.

3.6 Parameters

An overview of all public parameters and private parameters can be found in Tables 3 and 4 of Appendix A.

As discussed in Sect. 2, the parameter p and the number of iterations K are set to 0.85 and 50 respectively. The size of the RSA-modulus N is set to 2048 bit, as is recommended for 112-bit computational security [2], hence $N > 2^{2047}$. Moreover, the scaling parameters f_x and f_c are both chosen to be equal to $2^7 n$, where n is the total number of nodes in the graph. In Sect. 4 we will show that these scaling factors achieve a desirable level of precision.

Each PageRank value x_k^i is upper bounded by 1, and the total scaling factor after 50 iterations equals $(f_c)^{50} f_x = 2^{357} n^{51}$; the following condition will therefore guarantee that we do not encounter overflows:

$$n < 2^{33} \implies 2^{357} n^{51} < N.$$

In other words, the chosen parameter set returns approximated PageRank values for all graphs with less than 2^{33} nodes and there is no need to initialize the Damgård-Jurik cryptosystem with parameter $s > 1$.

For larger graphs a larger RSA-modulus N can be chosen or the Damgård-Jurik cryptosystem can be initialized with a larger exponent s. Another approach is the implementation of a secure division protocol, see for example [26], by which the size of the accumulated scaling factor, $(f_c)^k f_x$, can be reduced after some iterations.

4 Results

In this section, our secure multiparty PageRank solution is evaluated, both in terms of accuracy and in terms of computational and communication complexity. Firstly, the securely-computed PageRank values are compared to the standard PageRank values. Secondly, the running time of the algorithm is evaluated for various randomly-generated transaction graphs. Thirdly, the communication complexity of the protocol is analyzed, and finally, the results of the experiment are extrapolated to large-scale transaction graphs.

Our solution has been implemented in Python 3.5 using the General Multi-Precision library *gmpy2* and the Partially Homomorphic Encryption library *phe*.

All experiments were run in a single virtual machine with 48 CPU cores (2.4 GHz Intel Xeon E5-2680v4) and 16 GB RAM. The multiparty setting is emulated by assigning every core to exactly one party. The parties communicate by reading and writing data in a given folder on the virtual machine, which means the communication between the parties is performed instantly in this experimental set-up.

4.1 Accuracy of the Secure Multiparty PageRank Algorithm

According to Sect. 3.2 the results of the secure multiparty PageRank algorithm should be approximations of the standard PageRank values. For $i \in V$ let $s(i)$ be the standard PageRank value of node i and let $x(i)$ be the associated approximation computed by the secure multiparty PageRank algorithm. The accuracy of our solution can be quantified by various metrics. For the application in fraud detection, we are interested in point-wise comparison on the accuracy of secure PageRank. The maximum relative error satisfies this requirement and is computed as follows:

$$\max_{i \in V} \frac{|s(i) - x(i)|}{|s(i)|}.$$

To evaluate the accuracy of our algorithm, we randomly sampled directed graphs G, with n nodes and average out-degree d. To be more precise, for distinct nodes $i, j \in V$ we draw an edge from i to j with probability d/n. The nodes of this graph are distributed equally amongst 3 parties to represent the multiparty setting. Figure 1 displays the maximum relative error for randomly generated graphs with n ranging from 3×256 to 3×4096 and d ranging from 10 to 160.

The values $x(i)$ are computed by our implementation of the secure multiparty PageRank algorithm and the values $s(i)$ are computed with the PageRank functionality of the Python3 package NetworkX. These results show that the maximum relative error is between 0.0057 and 0.0064, meaning that our implementation indeed gives an accurate approximation of the standard PageRank values. For realistic graphs, it is more likely that large transaction graphs will approximate a scale-free graph [6]. Experiments on more realistic scale-free large graphs, with resulting PageRank values ranging from 10^{-6} to 10^{-2}, show that the maximum relative error stays under 0.006.

The accuracy of our algorithm is actually independent of the number of parties. To further increase it, the number of iterations K and/or the scaling factors f_x and f_c could be increased.

4.2 Performance of the Secure Multiparty PageRank Algorithm

To benchmark the computational complexity of our algorithm, we again consider randomly generated graphs with a total number of nodes n ranging from $|\mathcal{P}| \times 256$ to $|\mathcal{P}| \times 4096$ and an average out-degree d ranging from 10 to 160. Recall that $\mathcal{P}$ is the set of parties. In our first experiment, we fixed the number of parties to 3 and consider the computation time for various graph sizes. In our second experiment, we fix the average out-degree to 80 and vary the number of parties.

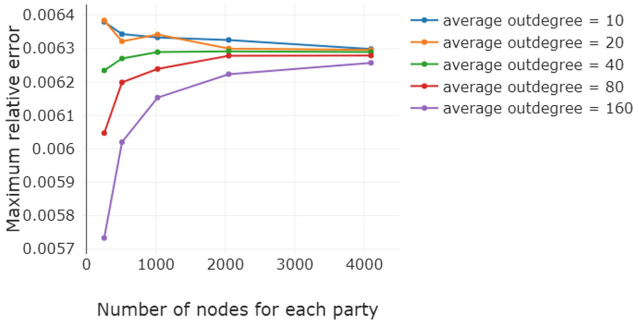

Fig. 1. Maximum relative error with increasing number of nodes and edges.

Note that the final step of our protocol, the decryption, is executed collaboratively, and therefore the computation times for each party are approximately equal. For this reason, only the computation time for party 1 is considered.

Graph Size. Figure 2 shows the results of our first experiment, in which the number of parties is fixed at 3 and each party runs their part of the protocol on a single core of the virtual machine. The computation time of the algorithm scales linearly in the number of nodes and in the average out-degree.

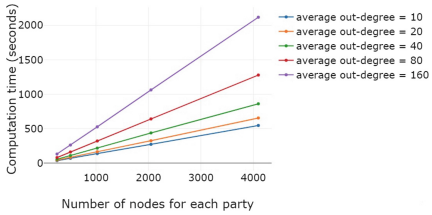

Fig. 2. Computation time for increasing numbers of nodes and edges.

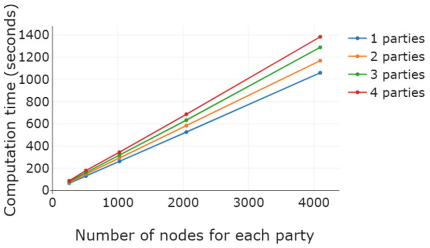

Fig. 3. Computation time for increasing numbers of parties.

Number of Parties. Figure 3 shows the results of our second experiment, in which the average out-degree is fixed to 80 and the number of parties is varied between 1 and 4. In the setting of our instantiations, each party is assigned the same number of nodes. Hence, the size of the total graph increases with the number of parties. The results show that increasing the number of parties only has a minor effect on the computational complexity of the protocol.

4.3 Parallelization

The secure multiparty PageRank algorithm is well-suited for distributed computing, an effect that we already observed when varying the number of parties. In our third experiment, the number of parties is fixed to 3 and each party owns 4096 nodes of a randomly-generated graph with an average out-degree of 80. Figure 4 shows the computation time of our algorithm when increasing the number of CPU cores per party. For this relatively small graph, a speed-up of factor 3.4 can be achieved by using 12 cores per party (instead of 1). By inspecting our algorithm, it is easily seen that larger graphs will benefit even more from distributing the computations over several CPU cores. Note that the upload respectively download communication times are excluded from this experiment (using 1 machine), but are expected to be within 5 respectively 9 s based on Table 1.

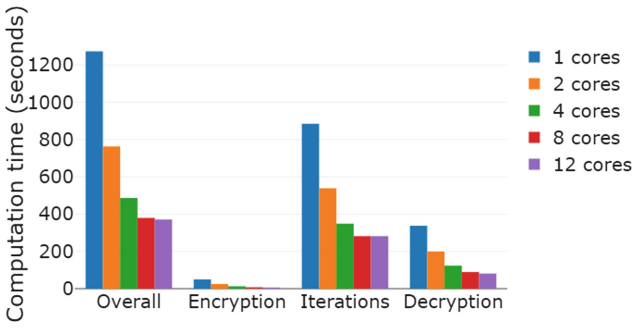

Fig. 4. Computation time for increasing numbers of cores.

4.4 Communication

The experiments of the above section were all run on a single machine, and the communication between the parties thus amounted to reading and writing data in a given folder. In real-life scenarios, the parties would be physically separated, and this data has to be communicated over some network. It is therefore important to measure the communication complexity of the protocol, both in terms of communication rounds and in terms of communicated data.

The number of communication rounds of our secure multiparty PageRank algorithm, excluding the key generation phase, equals $K + 1$, where K is the number of PageRank iterations. Table 1 displays the number of ciphertexts that have to be communicated in our protocol, under the assumption that the graph is randomly generated with average out-degree $d \gg |\mathcal{P}|$. Recall that $|\mathcal{P}|$ is the number of parties, n is the total number of nodes and n^P is the number of nodes of party P.

Table 1. The number of cipher texts that each party P uploads and downloads during the secure multiparty PageRank algorithm.

Phase	Number of uploaded cipher texts	Number of downloaded cipher texts
PageRank computation	$K(n^P + 1)$	$K(\|\mathcal{P}\| + n - n^P - 1)$
Decryption	n	$n - 2n^P + n^P\|\mathcal{P}\|$

Thus for a realistic-size 3-party setting with 10 million nodes per party and 50 PageRank iterations, Table 1 amounts to 530 million and 1040 million 2048-bit cipher texts that have to be uploaded and downloaded, respectively. Assuming a bandwidth of 100 Mbit/s, this results in 2.8 h upload time and 5.5 h download time.

The communication complexity is thus significant. However, it is currently not the main bottleneck as our experiments have shown that the required computation times are even larger. For this reason, we have not focused on improving the communication complexity. A first improvement could be made by noting that parties are not required to download the encrypted PageRank contributions of the entire graph for every iteration. Alternatively, another MPC paradigm with a much smaller communication complexity, such as fully homomorphic encryption [13], could be used. However, this would negatively impact the computation complexity.

4.5 Realistic Transaction Graphs

Based on experiments with scale-free graphs [6], it was shown that the maximum relative error of secure PageRank stays under 0.006.

The performance of our solution was evaluated on relatively small graphs, while in practice, transaction graphs contain millions of nodes. The first column of Table 2 shows the expected computation time for a 3-party setting in which each party has 10 million bank accounts with 80 transaction per bank account on average. Moreover, we assume a very conservative performance gain of a factor 3.4 by distributing the computation over 12 cores instead of using a single core per party; in practice, this gain will likely be much closer to a factor 12 for large graphs.

A significant improvement can be expected by judiciously implementing the algorithm in C++. In particular, 2048-bit modular multiplications have been reported to take only 3.012 ms on a Xeon X64 processor [7]. Modular multiplications take up a significant part of the computation time and can be accelerated up to a factor 17 compared to our Python implementation. The estimates in the second column of Table 2 show such an implementation would enable a running time of less than 1 day for a large-scale graph as described above.

Table 2. The estimated runtime for the application of the secure multiparty PageRank algorithm to a graph with 30 million nodes and average out-degree 80 in the 3-party setting.

	Python implementation	C++ implementation
Computation time (days)	10.62	∼0.62
Communication time (days)	0.35	0.35
Total time (days)	10.97	∼0.97

5 Conclusions and Future Research

Existing techniques for fraud detection would highly benefit from collaboration between financial institutes. However, the exchange of relevant information is often limited, or not even possible, due to privacy restrictions or commercial confidentiality. This paper illustrated that secure multiparty computation can help tackle this challenge.

It was previously shown that fraud detection techniques can be improved by taking into account features, such as PageRank values, derived from transaction graphs [20]. Transaction graphs of multiple financial institutes are coupled through interbank transactions and analyzing a combined transaction graph leads to a more complete picture and, possibly, to more effective fraud detection.

An innovative solution has been described for multiple financial institutes to securely and collaboratively compute the PageRank values of a combined transaction graph without revealing private and/or confidential information to each other or to any other party. Each financial institute learns the PageRank values of its own bank accounts that are derived from the joint network. To achieve the desired security properties an additively homomorphic encryption scheme has been used.

The feasibility of this secure multiparty PageRank algorithm has been shown by implementing it in Python and applying it successfully on randomly generated test networks while simulating up to four different parties. Furthermore, the results show that the secure multiparty PageRank algorithm is scalable and can handle transaction graphs of realistic size in reasonable time, while remaining sufficiently accurate. In particular, this allows financial institutes to update PageRank scores at reasonable intervals, e.g. over a month, and use these as features for their own fraud detection methods.

Planned future work includes developing, in contrast to the current passively secure protocol, an actively secure solution. In addition, other homomorphic encryption schemes, in particular ones that can withstand the attack of a quantum computer, will be considered.

Moreover, our MPC solution is tailor made to evaluate the PageRank algorithm securely as it specifically utilizes the linearity of this algorithm. However, this linearity applies to many other algorithms and an analysis of the broader applicability of this MPC approach would be of interest. In particular since

PageRank is only one of many graph-based features that may be extracted from transaction graphs as input for fraud detection methods. In addition, there are other application domains in the financial sector that could benefit from the secure analysis of transaction graphs, such as commercial banking and anti-money laundering. Further important future work is testing the algorithm on actual combined transaction graphs and analyzing its effect on the resulting PageRank values and its effectiveness in improving fraud detection.

In conclusion, the feasibility of securely analyzing features of a large-scale network that is distributed over multiple parties has been demonstrated, thus paving the way for several collaboration initiatives that were previously not possible.

Acknowledgments. The research activities that have led to this paper were funded by the Shared Research Program Cyber Security; a research collaboration between TNO, ABN AMRO, Rabobank, ING, Achmea and Volksbank. The authors would also like to thank Gabriele Spini for his valuable feedback and his help in improving the paper.

A Parameters

The following public and private parameters are considered in the secure PageRank algorithm, Algorithm 1.

Table 3. Public parameters of the secure multiparty PageRank algorithm

Parameter	Description
$\mathcal{P}$	Set of parties
n	Number of nodes
N	RSA modulus
K	Number of PageRank iterations
p	PageRank probability
f_c	Scaling factor for all $\frac{p}{c_i}$
f_x	Scaling factor for all x_k^i
ϕ_k	Integer PageRank constant of iteration k
Φ_k	Encrypted PageRank constant of iteration k
ψ	Integer PageRank constant
pk	Public Damgård-Jurik encryption key
z_k^i	Encrypted and scaled PageRank value of node i at iteration k

Table 4. Private parameters of the secure multiparty PageRank algorithm

Parameter	Description
V^P	Set of all nodes belonging to party P
U^P	Set of non-dangling nodes belonging to party P
D^P	Set of dangling nodes belonging to party P
$S(i)$	Set of incoming nodes at node $i \in V$
c_i	The out-degree of node $i \in V$

References

1. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Van Vyve, M.: Securely solving simple combinatorial graph problems. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 239–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_21
2. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M., Zieglar, L.: Recommendation for key management - part 1: General (revision 4). National Institute of Standards and Technology - Special Publication **800**(57), 1–156 (2015)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 2–4 May 1988, Chicago, Illinois, USA, pp. 1–10. ACM (1988). https://doi.org/10.1145/62212.62213
4. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88313-5_13
5. Bogetoft, P., et al.: Secure multiparty computation goes live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03549-4_20
6. Bollobás, B., Borgs, C., Chayes, J., Riordan, O.: Directed scale-free graphs. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2003, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 132–139 (2003). http://dl.acm.org/citation.cfm?id=644108.644133
7. Bos, J.W., Montgomery, P.L., Shumow, D., Zaverucha, G.M.: Montgomery multiplication using vector instructions. IACR Cryptology ePrint Archive, vol. 2013, p. 519 (2013). http://eprint.iacr.org/2013/519
8. Brickell, J., Shmatikov, V.: Privacy-preserving graph algorithms in the semi-honest model. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 236–252. Springer, Heidelberg (2005). https://doi.org/10.1007/11593447_13
9. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 2–4 May 1988, Chicago, Illinois, USA, pp. 11–19. ACM (1988). https://doi.org/10.1145/62212.62214
10. Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential benchmarking based on multiparty computation. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 169–187. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_10

11. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_9
12. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Proceedings of Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2012, pp. 643–662 (2012). https://doi.org/10.1007/978-3-642-32009-5_38
13. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May – 2 June 2009, pp. 169–178. ACM (2009). https://doi.org/10.1145/1536414.1536440
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing 1987, New York, USA, pp. 218–229 (1987). https://doi.org/10.1145/28395.28420
15. Haveliwala, T., Kamvar, S.: The second eigenvalue of the google matrix. Tech. rep. 2003–20, Stanford InfoLab (2003). http://ilpubs.stanford.edu:8090/582/
16. Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T.: Efficient RSA key generation and threshold Paillier in the two-party setting. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 313–331. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_20
17. Henecka, W., Kögl, S., Sadeghi, A., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, 4–8 October 2010, pp. 451–462. ACM (2010). https://doi.org/10.1145/1866307.1866358
18. Ipsen, I.C.F., Selee, T.M.: Pagerank computation, with special attention to dangling nodes. SIAM J. Matrix Anal. Appl. **29**(4), 1281–1296 (2007). https://doi.org/10.1137/060664331
19. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: Blaze, M. (ed.) Proceedings of the 13th USENIX Security Symposium, 9–13 August 2004, San Diego, CA, USA, pp. 287–302. USENIX (2004). http://www.usenix.org/publications/library/proceedings/sec04/tech/malkhi.html
20. Molloy, I., et al.: Graph analytics for real-time scoring of cross-channel transactional fraud. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 22–40. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_2
21. Nayak, K., Wang, X.S., Ioannidis, S., Weinsberg, U., Taft, N., Shi, E.: GraphSC: parallel secure computation made easy. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 377–394. IEEE (2015)
22. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_40
23. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
24. Partisia: Secure order matching (2018). https://partisia.com/order-matching/. Accessed 31 July 2018

25. Unbound: Hybrid cloud key management for any key, any cloud (2018). https://www.unboundtech.com/usecase/hybrid-it-key-management-any-key-any-cloud/. Accessed 31 Jul 2018
26. Veugen, T.: Encrypted integer division and secure comparison. IJACT **3**(2), 166–180 (2014). https://doi.org/10.1504/IJACT.2014.062738
27. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3–5 November 1982, pp. 160–164. IEEE Computer Society (1982). https://doi.org/10.1109/SFCS.1982.38
28. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27–29 October 1986, pp. 162–167. IEEE Computer Society (1986). https://doi.org/10.1109/SFCS.1986.25