PrivateGraph: A Cloud-Centric System for Spectral Analysis of Large Encrypted Graphs

Sagar Sharma and Keke Chen
Data Intensive Analysis and Computing (DIAC) Lab, Kno.e.sis Center
Department of Computer Science and Engineering, Wright State University, Dayton, OH
Email: {$sharma.74, keke.chen$}@wright.edu

***Abstract*—Graph datasets have invaluable use in business applications and scientific research. Because of the growing size and dynamically changing nature of graphs, graph data owners may want to use public cloud infrastructures to store, process, and achieve graph analytics. However, when outsourcing data and computation, data owners are at burden to develop methods to preserve data privacy and data ownership from curious cloud providers. This demonstration exhibits a prototype system for privacy-preserving spectral analysis framework for large graphs in public clouds (PrivateGraph) that allows data owners to collect graph data from data contributors, and store and conduct secure graph spectral analysis in the cloud with preserved privacy and ownership. This demo system lets audience to interactively learn the major cloud-client interaction protocols: the privacy-preserving data submission, the secure Lanczos and Nyström approximate eigen-decomposition algorithms that work over encrypted data, and the outcome of an important application of spectral analysis - spectral clustering. In the process of demonstration the audience will understand the intrinsic relationship amongst costs, result quality, privacy, and scalability of the framework.**

I. Introduction

From social networks, mobile, and web applications, to biomedical research, large graphs have become an important data source, providing great value in both business and scientific research. Particularly, spectral analysis of graphs gives important results pertinent to community detection, PageRank, and spectral clustering. Despite their utility, their size and dynamic nature make large graphs expensive to store, maintain, and analyze. Hence, data owners may rely on cloud services for storage and analysis so as to eliminate the need to establish, operate, and maintain expensive in-house infrastructures.

However, outsourcing storage and analytics to cloud brings serious concerns of data privacy and ownership. External adversaries and curious employees at the cloud provider can potentially gain unauthorized accesses to outsourced data and intermediate computation results [3], [7]. It is hence crucial to investigate privacy-preserving mechanisms for outsourcing data and analytics to the cloud.

Two common generic privacy preserving approaches: fully homomorphic encryption (FHE) and secure multi-party garbled circuits (GC) [4] can theoretically construct the privacy-preserving versions of most data mining algorithms, including spectral analysis. Nevertheless, they are too expensive to be practical. The current best implementation of FHE schemes [2] results in large ciphertext (e.g., a 4-byte integer will become about 100 kilobytes - 25,000 times of size increase) and expensive homomorphic multiplication (e.g., about 10 milliseconds for a single multiplication). On another example, the privacy-preserving matrix factorization solution for recommendation systems [5] using garbled circuits incurs 40 gigabytes of communication cost in one iteration of factoring a small-size 100×100 matrix.

We recently proposed the privacy-preserving spectral analysis framework for large graphs in public clouds [8], a cloud-centric framework that enables data owners to privately store and analyze large graph datasets with untrusted public cloud. Distributed data contributors contribute to a graph database in the cloud with encrypted submissions using a certain additive homomorphic encryption (AHE) scheme, such as Paillier scheme [6] and pairing scheme [1] that have much lower costs than the current FHE schemes. The data owner interacts with the cloud to run one of the two privacy-preserving graph spectral analysis algorithms to get the top-k eigenvectors.

Specifically, the framework includes a privacy-preserving distributed sparse graph data submission algorithm based on differential privacy that allows data contributors to attain personalized privacy while maintaining graph sparsity. It uses *pseudo homomorphic multiplication* and random data masking mechanisms to enable cloud-side homomorphic matrix computations essential to constructing the secure Lanczos and the secure Nyström algorithms for the core operation: eigendecomposition. For a N-node graph, the cloud-client collaborative algorithms we developed ensure all operations of complexity $O(N^2)$ happen in the cloud with preserved scalability, while only those computations of complexity $O(N)$ fall upon client's responsibility. The communication costs also remain $O(N)$ to make the whole framework practical.

To help readers fully understand the ideas behind the framework, we develop the PrivateGraph interactive demonstration system that consists of the following components: 1) the user interface for selecting/adding/updating graph data, and setting sparsity and privacy parameters for data submission, 2) the user interface for setting up and running the secure Lanczos and Nyström algorithms, 3) the implementation of the core algorithms in both the cloud and the client sides, 4) the visualization of the result of spectral clustering, and 5) the statistics of the storage, computation, and communication costs for the involved parties.

II. PrivateGraph Architecture

Figure 1 shows the PrivateGraph framework and the involved three parties: the public cloud (server), the data owner (client), and data contributors. The data owner periodically runs graph algorithms on the evolving graph datasets to generate models. It also generates the public-private key pair and distributes the public key to the data contributors. Data contributors upload encrypted graph data (i.e., edges) to the cloud through secure interfaces. Alternatively, the data owner can also directly upload to the cloud any in-house graph datasets collected via other channels. The cloud and data owner collaboratively run the secure eigen-approximation algorithms over encrypted dataset residing in the cloud without compromising privacy. The expensive operations $O(N^2)$ are carried out by the cloud while the lighter computations $O(N)$ by the client.

Threat Model: Data contributors' privacy, data-ownership, and privacy of data in between computations are the assets at stake [10]. PrivateGraph preserves these assets from "honest but curious" cloud without compromising the utility of graph datasets.

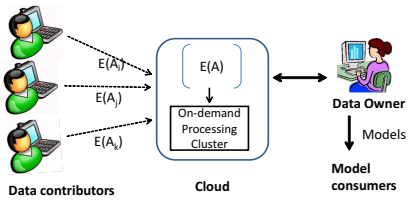

Fig. 1: The PrivateGraph framework. Distributed users contribute to the graph data which is encrypted and stored in public cloud. The graph spectral analytics happen with cloud-data owner collaboration.

A. Sparse data submission

In order to use the AHE schemes for encryption the message space is converted to *non-negative integers*. For matrices in the real value domain, we apply a reversible affine transformation which preserves d decimal places: For a real value α,

$$g(\alpha) = 10^d \alpha + \tau, \tag{1}$$

where τ is a positive value large enough to convert the smallest negative value to a non-negative.

Let us work on the spectral analysis problem for the *normalized graph Laplacian matrix* that can be used for spectral clustering [9]. Each data contributor represents one or a few nodes in the graph, and contributes to the corresponding row(s) of this graph matrix.

For an undirected graph, let D be the diagonal matrix with node degrees on its diagonal - D_{ii} represents the degree of node i, $i = 1..N$. Let W be the adjacency matrix such that W_{ij} =1 if and only if the edge (i, j) exists, and $W_{ij} = 0$ otherwise. For undirected graphs, W is a symmetric matrix, where each row(column) of W represents the corresponding node's adjacency edges. The normalized graph Laplacian matrix L is $L = I - D^{-1}W$, where I is the N by N identity matrix. Specifically, for each row of the Laplacian matrix, say L_i, its element $l_{ij}, j = 1..N$ is:

$$l_{ij} = \begin{cases} -1/D_{ii} & \text{if } W_{ij} = 1 \text{ and } i \neq j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Encoding and encrypting the entire vector in a dense format is straightforward, i.e. convert each element of the row with Eq. 1 and then encrypt with the AHE scheme. Next, we focus on the sparse encoding and encryption of matrix for its cost-saving benefit.

Most graphs in popular applications are sparse. It is desirable to skip encrypting some of the zero entries to save storage, communication, and computation costs, while still providing sufficient protection to privacy. We design an algorithm that protects the privacy of node degrees and links using differential privacy while preserving data sparsity and authenticity.

First, we transform the Laplacian matrix L without affecting its eigen structure. Let γ be a sufficiently large positive integer such that $\lfloor \gamma / D_{ii} \rfloor$, where D_{ii} is the degree of node i, for all $i = 1..N$, are also positive integers with necessary precision preserved. Let $H = \gamma(I - L)$, which converts entries to positive and preserves the sparsity of L. Let *top-k (or bottom-k) eigenvectors* of a graph matrix be the eigenvectors corresponding to the largest (or smallest) k eigenvalues, $k = 1..N$. We have the following Proposition, the proof for which is included in the original paper [8].

Proposition 1: The top-k eigenvectors of H are the same as the bottom-k eigenvectors of L.

Note that the bottom-k eigenvectors of L are used for spectral clustering [9].

With the above transformation and proposition, we present our bin-based graph disguising algorithm as Algorithm 1. Each node (data contributor) injects the fake edges (i.e., the encrypted zero entries in the matrix) into the edge submission, so that the differential privacy is satisfied to disguise node degrees and the existence of edge, which are critical to existing privacy attacks [10]. The number of noisy edges is affected by the bin that the node degree falls in, so that the sparsity is better preserved. Note that these encrypted zero entries cannot be distinguished from other entries and keep the authenticity of the spectral analysis results.

Algorithm 1 Privacy preserving sparse submission (H, ϵ, $d_{i,j}$).

1: input: B: histogram provided by the data owner. ϵ: user selected parameter for ϵ-differential privacy. $d_{i,j}$: the actual node degree.
2: find the bin that contains $d_{i,j}$, whose upper bound and lower bound are U_i and L_i, respectively;
3: $b \leftarrow (U_i - L_i)/\epsilon$;
4: $q \leftarrow b * 3.912$;// for $b = 1$ the q value is at 3.912, which scales with b;
5: draw a value $\delta_{i,j}$ from the distribution Laplace $(0, b)$;
6: $k_{i,j} \leftarrow |q| + \delta_{i,j}$;
7: add the $d_{i,j}$ real links to the list with the sparse encoding;
8: randomly choose $k_{i,j}$ edges from the rest $N - d_{i,j}$ edges and encode them as the encrypted zero entries;
9: submit the $d_{i,j} + k_{i,j}$ items.

B. Privacy-preserving Lanczos algorithm

The Lanczos algorithm is an iterative method for finding the top-k eigenvectors. In the following, we let the matrix A represent any graph matrix including the previously defined matrix

H. The Lanczos iterations for a matrix $A_{N\times N}$ start with a random N-dimensional vector b_0, and the most expensive operation in each iteration - the matrix-vector multiplication $b_i = Ab_{i-1}$.

The secure Lanczos algorithm consists of two key ideas. (1)The pseudo homomorphic matrix-vector multiplication $E(A\hat{b}_{i-1}) = f(E(A), \hat{b}_{i+1})$ in the cloud, working with the encrypted graph matrix $E(A)$ and the encrypted vector $\hat{b}_{i-1}$ which is a perturbation of the original vector b_{i-1}. (2) The cost-effective algorithms for generating the perturbed $\hat{b}_{i-1}$ from b_{i-1} and recovering Ab_{i-1} from $A\hat{b}_{i-1}$ in $O(N)$. These algorithms provide strong security guarantee that $\hat{b}_{i-1}$ cannot be distinguished from any vector sampled uniformly at random.

The cloud-side pseudo homomorphic computation of $E(A\bar{b}_{i-1})$ can be easily cast to a matrix-row based parallel algorithm such as MapReduce and Spark.

Algorithm 2 Privacy-preserving Lanczos Algorithm with AHE schemes

1: $b_0 \leftarrow$ random N-dimensional vector and encrypt $E(b_0)$; // data owner
2: download $E(b_0)$, compute $E(A_i b_0)$, and send back to data owner; // each data contributor
for $i \leftarrow 1$ to t do:
3: $\bar{b}_{i-1} \leftarrow$ perturbation based on $\{b_0, .., b_{i-2}\}$ and seed vectors $\{s_1, .., s_h\}$ and upload $\bar{b}_{i-1}$; // data owner
4: compute $E(A\bar{b}_{i-1})$; // cloud
5: download and decrypt $E(A\bar{b}_{i-1})$; // data owner
6: recover b_i from $A\bar{b}_{i-1}$; // data owner
7: $\alpha_{i-1} \leftarrow b_i^T b_{i-1}$; // data owner
8: $w_{i-1} \leftarrow b_i - \alpha_i b_{i-1} - \beta_{i-1} b_{i-2}$ where $b_i = 0$ for $i < 0$; // data owner
9: $\beta_i \leftarrow \|w_{i-1}\|$; // data owner
10: $b_i \leftarrow w_{i-1}/\beta_i$; // data owner
11: end for
12: Decompose the trigonal matrix $T_{t\times t}$ consisting of $\{\alpha_i\}$ and $\{\beta_i\}$ to get the eigenvectors.// data owner

C. Privacy-preserving Nyström Algorithm

In a plain setting, the cloud subsamples the matrices $C_{N\times m}$ and $W_{m\times m}$ from the entire matrix $A_{N\times N}$, and sends W to the data owner. The data owner decomposes W and sends back the result $U_{m\times k}$ and $\Lambda_{k\times k}$. Finally, the cloud computes the result $V = U_{m\times k}\Lambda^{-1}_{k\times k}$ and CV, which is sent to the data owner. The privacy problem with above steps is that the cloud-side computation of $CU_{m\times k}\Lambda^{-1}_{k\times k}$ requires $U_{m\times k}$ and $\Lambda_{k\times k}$ in plaintext, which reveals the eigen-structure of W and thus breaches privacy.

The privacy of V is achieved by a disguising and recovering protocol that is similar to the one we use for the Lanczos algorithm. Specifically, there are three key aspects of the protocol. (1) Data owner will submit a masked V matrix: $\bar{V} = V + \Delta \mod q$, so that $\bar{V}$ cannot be distinguished from any uniformly drawn random matrix, which can be formally proved. (2) The expected result CV can be recovered at a cost $O(kN)$ from $C\bar{V}$ that is provided by the cloud with the pseudo homomorphic multiplication $E(C\bar{V}) = E(C)\bar{V}$. (3) The data owner only needs to download $O(kN)$ encrypted data. The demonstrated system will show the detail of generating the Δ matrix and the auxiliary data that the cloud and the data contributors need to compute.

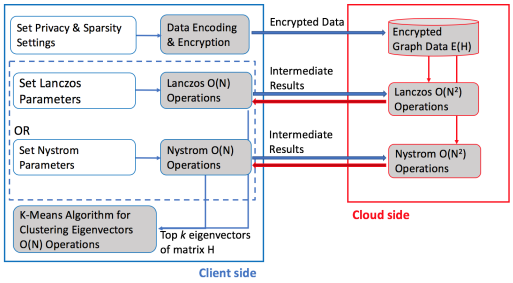

Fig. 2: The PrivateGraph workflow. The gray boxes represent the back-end while the white boxes are client facing.

III. Demonstration

This demonstration is intended to give an interactive experience for the audience to understand the PrivateGraph framework. Users of the demo system get to select/create/update graph datasets, tune the data submission parameters including privacy and sparsity setting, select the eigen-approximation method and related parameters, and finally observe the spectral clustering results with cluster visualization, and check the statistical summary. Several synthetic small graph datasets will be provided for users to play with the live system. This demonstration consists of following two parts.

Introduction. A poster will introduce the PrivateGraph framework, explain its components, and summarize its performance with real graph datasets. The poster will also identify problems and challenges with related existing methods and how PrivateGraph uniquely addresses those problems.

Live System. A fully interactive implementation of PrivateGraph framework will be presented to the audience. One can interact with the client program and perform following tasks: 1) Create/update graph datasets, 2) Tune the sparsity and privacy parameters for data submission, 3) Submit the graph data to the cloud server, 4) Select eigen-approximation algorithm and set its parameters, 6) Receive eigenvectors and visualize the result of spectral clustering, and 7) Observe the statistical measures and the cost distribution between cloud and client.

A. Demonstration workflow

The audience follows the step-by-step workflow during the demonstration after a guided explanation of the poster outlined in Figure 2.

1) Select one of the existing graph datasets, or start with an editable custom graph.

2) Set privacy and sparsity parameters. Then submit the graph data to the cloud provider.

3) Pick either secure Lanczos or secure Nyström method for eigen-approximation and set corresponding parameters. For Lanczos, specify the number of iterations and number of clusters required for spectral clustering. For Nyström, specify the sampling rate and number of clusters.

4) Wait for the cloud-client protocol to finish the eigen-approximation steps.

5) Conduct graph spectral clustering on the obtained eigenvectors in the client.

6) Observe the visualization of the graph clustering result and the statistical summary outlining the cost distribution between client and cloud.

B. Live System

Our live system consists of these major components: 1) the fully interactive front-end user interface or the client program for parameter settings and issuing commands; 2) the data sparse encoding and encryption system, 3) the cloud-client Lanczos and Nyström algorithms, 4) the K-means algorithm for clustering eigenvectors, 5) the cluster visualization and statistics summarization tool.

The client-side interface allows audience to create/update graph datasets and performs task described in step one of the demonstration work-flow in section III-A. Figure 3 gives the wireframe for the client interactive program.

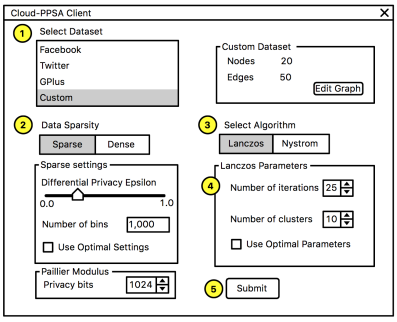

Fig. 3: PrivateGraph Client-side Interface. Users can select preset datasets or add and modify their own datasets, and tune data submission and eigenapproximation parameters.

In the background, the data encoding and encryption system will transform graph data to a matrix and encrypt it with desired Paillier security level. The differential privacy parameter, ϵ, is used to determine the level of sparsity and generates the fake random edges. Each of the privacy-preserving algorithms of Lanczos and Nyström operates collaboratively between client and cloud. The Lanczos method incurs several rounds of communication between client and cloud while Nyström method incurs one round of communication. The client side will need to decrypt the intermediate results during the computations.

We use graph spectral clustering to illustrate the effectiveness of the algorithms. After getting the top-k eigenvectors of the H matrix, the client side locally applies the K-means algorithm on the eigenvectors to determine the clusters. The visualization tool is used to graph the clusters and present it to the audience. The result quality is computed by the tool and visually verified by the audience. The live system will also keep track of computation and communication cost between client and cloud and summarize the result in the end. By tuning the parameter settings in the Lanczos or Nyström algorithm, users can observe their effect on the clustering results and also the costs incurred in both the cloud and the client. Thus, the audience will get a good understanding of the intricate relationship between cost and result quality. Fig 4 shows the output screen shown to the audience upon completion of the workflow.

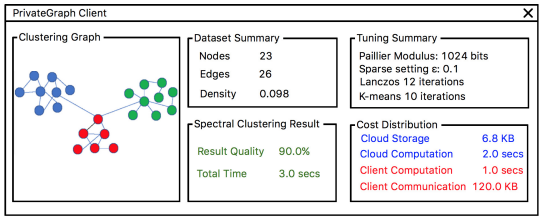

Fig. 4: PrivateGraph Client-side Result Screen. Users get a visualized clustering result, summary of result quality and distribution of cost.

IV. Summary

The PrivateGraph demonstration showcases the key components of the published privacy-preserving spectral analysis framework for large graphs in public clouds [8], which successfully balances privacy and utility in outsourced graph analysis. The demonstration provides the audience with background knowledge of the privacy-problem in outsourced data analytics and then exhibit the proposed solution for graph spectral analysis with a poster, a presentation, and a live system. The audience will partake in the data submission mechanism, tune the privacy level and data sparsity, select an eigen-approximation method and related parameters, and observe the spectral clustering results including cluster visualization, clustering quality, and the cost distribution between cloud and client.

References

[1] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.

[2] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[3] A. Chen. Gcreep: Google engineer stalked teens, spied on chats. *Gawker, http://gawker.com/5637234/*, 2010.

[4] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 35–35, Berkeley, CA, USA, 2011. USENIX Association.

[5] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, pages 801–812, New York, NY, USA, 2013. ACM.

[6] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238. Springer-Verlag, 1999.

[7] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[8] S. Sharma, J. Powers, and K. Chen. Privacy-preserving spectral analysis of large graphs in public clouds. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pages 71–82, New York, NY, USA, 2016. ACM.

[9] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[10] B. Zhou, J. Pei, and W. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor. Newsl.*, 10(2):12–22, Dec. 2008.