A Minimum Path Calculation Algorithm for Protecting the Edge Weight Privacy in Cloud Circumstance

Xin LIU¹,a, Huan Li²,b

¹School of Network Engineering, Zhoukou Normal University, Henan, Zhoukou, 466001, China
²School of Computer Science and Technology, Zhoukou Normal University, Henan, Zhoukou, 466001, China

aliux_znu@163.com, bliuxin_2017@126.com
*Corresponding author

Keywords: Cloud Computing, Shortest Distance Computing, Encrypted Graph Data, Privacy-preserving, Homomorphic Encryption.

Abstract. Based on the encryption algorithm, this paper presents an algorithm to calculate a minimum path between nodes, which protects the privacy and security of edge weights. This algorithm carries on the encryption to the edge weight information, and then the encrypted data is outsourced to the cloud platform. Using breadth first theory, cloud server can directly achieve the graph like data encryption from the initial node outward, and calculate the minimum path from the initial node to the other nodes. This aims to calculate the minimum path between any two nodes. The minimum path calculated by the client and cloud server interaction in collaboration. The main calculation can be from the server, the client only decrypt and judges the operation simple. This algorithm can make the low computational capability of the client and also use cloud computing task to calculate the minimum path computation intensive, and to ensure that the intermediate results of calculation of edge weights. The final result of privacy information will not be leaked. Theoretical analysis and simulation experiments show the safety and correctness of the algorithm.

Introduction

In recent years, the figure of the graph data in the real world has been widely used in lots of applications[1]. Calculation of the shortest distance between any two nodes in graph data is one of the basic functions in data processing. More and more individuals and companies choose to calculate the shortest distance in the cloud computing environment in order to save local resource. However, the user has lost direct control of its data and does not know the real situation of data processing in the cloud environment. From the user point of view, the cloud computing environment is not credible and unsafe[2]. If cannot guarantee the privacy of data information, the user will not rent the cloud server to calculate the shortest distance between the graph data nodes on a large scale and it will seriously hinder the use and development of cloud computing. A series of calculation methods for protecting data privacy are proposed in the literature[3-5]. However, the existing research can either protect the privacy of a specific information or impractical. The edge weights information involved in the shortest distance calculation in cloud computing environment was not effectively protected. Therefore, in order to solve the issue of edge weight information privacy protection when calculating the shortest distance between nodes in cloud computing environment, an algorithm used to calculate the shortest distance between nodes based on homomorphic encryption (HESDC) is proposed. HESDC uses the encrypted graph data in the cloud platform to calculate the shortest distance so that the real edge weight information will not be illegally obtained by the cloud service provider or unauthorized users, which effectively guarantee the security of the user's privacy.
Problem Description

System Model and the Threat Model

Cloud service providers provide users with a powerful storage and calculation capacity. User's local resources are limited so they need to rent storage and calculation resources from the cloud service providers, which is used to calculate the shortest distance between the nodes for a large scale of graph data. Users first use Paillier encryption algorithm 0 to encrypt the edge weights of the graph data, and then outsource it to the cloud environment. For the encrypted data, the cloud server and client use the HESDC algorithm to calculate the shortest distance between nodes.

In the cloud environment, the user's graph data faces a great risk of privacy leakage. On the one hand, due to the virtualization, cloud computing technology such as multi-tenancy is unsafe. Malicious users in the cloud environment or the attacker use the existing security risks to illegal access to user's data and privacy. On the other hand, after the user's graph data outsourcing to the cloud environment, the user has lost the direct control of its data information and does not know the real processing situation of his data which contain abundant of valuable information. Privileged users in the cloud environment even the cloud computing providers themselves guarantee not illegal to steal user information, but may be for some specific purpose, they use the actual control of physical resources and data in the process of performing calculations and legally collect, statistics and analysis user data which violate the user's privacy.

The Background of Shortest Distance Calculation

The shortest distance calculation is one of the basic graph data processing functions, and it is defined as follows.

**Definition 1.** (Shortest distance calculation) For a graph data \( G(V,E) \), Where \( V \) and \( E \) are the nodes and edges of the graph data. Note \( e=(u,v) \) for the edge of connecting of nodes \( u \) and \( v \), its weight is \( w(e) \) or \( w(u,v) \), the weight is assumed to be a non negative integer and if \( u \) and \( v \) are not connected, the weight is infinity. The path between the two nodes is composed of a series of connected components, denoted as \( P \), the set of paths is denoted as \( P \). The weight of the path is the sum of the weights of all edges on the path, denotes as \( len(P) \). The shortest distance between the two nodes is the minimum value of the weight of all paths, where

\[
\delta(u,v) = \min \left\{ len(P) = \sum_{e \in P} w(e) \mid P \in P \right\}.
\]  

(1)

When the graph data is outsourced to the cloud computing environment for the shortest distance calculation, it is encrypted in order to protect the edge weight information; formula (1) is transformed to

\[
E_k(\delta(u,v)) = \min \left\{ E_k(len(P)) = \sum_{e \in P} E_k(w(e)) \mid P \in P \right\},
\]

(2)

where \( E_k \) is the Paillier encryption function before the graph data is outsourced to the cloud computing environment in this paper.

Method for Calculating the Shortest Distance with Edge Weight Protecting

HESDC includes two stages of preprocessing and computing.

Preprocessing stage: The user first uses a random number generation algorithm to generate the Paillier encryption algorithm, which is required by the relevant key. Then the weight information of the edge is encrypted to get the encrypted data \( G_k \), and finally \( G_k \) is outsourced to the cloud computing environment.
Computing stage: The cloud server uses the proposed HESDC algorithm in this paper to calculate the shortest distance between any two nodes in the encrypted graph data. The whole calculation process is completed by the cloud server and client interaction. The main calculation is performed in the cloud computing environment, the user only need to carry out a simple decryption and judgment operation, and record the results of the calculation.

The edge weights of the graph data is encrypted by $E_k(w(u,v))$ and the HESDC calculates the encrypted data, the basic principle is similar to the Dijkstra algorithm, which is interactively calculation of the shortest distance between the initial node and the other nodes. Assume that the initial node is $s$, the destination node is $d$, and the main calculation flows are as follows.

Step 1. The distance between node $s$ to the other nodes is marked as infinite, and all the other nodes are marked as undetermined. The initial node $s$ is set to the current node $u$.

Step 2. Cloud server calculate the distance of initial node $s$ to node $v$, which is marked for the uncertain neighbor node of current node $u$, calculate as

$$E_k(d) = E_k(dis(s,u))E_k(\omega(u,v)).$$

Step 3. Cloud server return $E_k(d)$ to the client to decrypt it, and compared with $dis(s,v)$, which is the calculated distance of $s$ and $v$. If $d$ is smaller, then update $dis(s,v) = d$ and its encrypted value $E_k(dis(s,v)) = E_k(d)$.

Step 4. All the neighbor nodes in the node $u$ have been calculated and labeled as known nodes.

Step 5. Select the closest nodes from the unknown nodes to the original node $s$ as the new current quarter point $u$.

Step 6 Repeat steps 2 to 5, until the destination node $d$ is marked as known, that is, the shortest distance of $s$ and $d$ is obtained.

This algorithm can also be used to calculate the shortest distance of initial node to all other nodes, that is, when all the nodes are labeled as known, the algorithm stops. The calculation of HESDC in the cloud server is based on the Encrypted information of the edge weights. Paillier encryption algorithm is semantic security and its key is securely held by the user. As long as the user’s key is not access to other users, you can guarantee the graph data is not attacked and the privacy of information is security, so the information of edge weight is not leak to any unauthorized users or illegally obtained by other attackers.

Performance Evaluation and Analysis

We evaluate the local computing resources and computing time use the HESDC algorithm by simulation experiments to verify the validity and usability of calculating the shortest distance in a large scale of nodes by using HESDC for lower local resources. The specific settings of the experiment: suppose $n$ and $m$ are the number of nodes and edges. In this paper, we randomly select $m$ source nodes and destination nodes from $n$ nodes to generate the graph data, Edge weights are randomly selected from the [1,100] and the average number of nodes is 3, the experimental nodes are selected as 500, 1000 and 1500 respectively. The key length of the Paillier encryption algorithm is set to 512 bit. The machine configuration of the simulated cloud server and the local server are dual core Intel processors (2.93GHz * 2).

In the preprocessing stage, it is necessary to encrypt the edge weights of the graph data, which will bring extra cost. As shown in Figure 1, where: $M$ for the node size; $Q$ for the storage space; $t$ for the time required to add a secret, Encryption increases the storage consumption of graph data and requires a certain processing time. The storage consumption and the encryption time increase approximately linearly with the node size of the graph data.
Figure 1. The Storage and Time Consumption of the Algorithm Proposed in this Paper.

Memory ratio $r$ is defined as the ratio of locally required memory of HESDC and Dijkstra algorithm. As shown in Figure 2, with the increase in the size of the graph data, the memory ratio decreases dramatically. When the node size increases to 1500, the proposed algorithm required only 2.45% memory resources of the Dijkstra algorithm. When further increasing scale of node, memory rate will reduce. Therefore, HESDC can effectively reduce the memory resource cost when calculating shortest distance between nodes in large scale graph data, which makes weaker computers calculating the shortest distance between large scales of graph data nodes possible by using the cloud computing technology.

References


