Secure Error-Tolerant Graph Matching Protocols

Kalikinkar Mandal[1], Basel Alomair[2], and Radha Poovendran[1]

[1]Network Security Lab, Department of Electrical Engineering, University of Washington, Seattle, WA, 98195, USA.
[2]National Center for Cybersecurity Technologies, King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia. Email:{kmandal,alomair,rp3}@uw.edu

Abstract. We consider a setting where there are two parties, each party holds a private graph and they wish to jointly compute the structural dissimilarity between two graphs without revealing any information about their private input graph. Graph edit distance (GED) is a widely accepted metric for measuring the dissimilarity of graphs. It measures the minimum cost for transforming one graph into the other graph by applying graph edit operations. In this paper we present a framework for securely computing approximated GED and as an example, present a protocol based on threshold additive homomorphic encryption scheme. We develop several new sub-protocols such as private maximum computation and optimal assignment protocols to construct the main protocol. We show that our protocols are secure against semi-honest adversaries. The asymptotic complexity of the protocol is $O(n^5\ell \log^*(\ell))$ where ℓ is the bit length of ring elements and n is the number of nodes in the graph.

1 Introduction

Graph matching is a task of assessing the structural similarity of graphs. There are two types of graph matching, namely *exact matching* and *error-tolerant matching* (also known as inexact matching) [1, 26, 29]. The exact graph matching aims to determine, whether two graphs – a source graph and a target graph – are identical. The later one aims to find a distortion or dissimilarity between two graphs. Graph edit distance is a metric that measures the structural dissimilarity between two graphs. The graph edit distance is quantified as the minimum costs of edit operations required to transform the source graph into the target graph. We consider an attribute graph consisting of a set of nodes, a set of edges and labels assigned to nodes and edges. Examples of such graphs are social network graphs and fingerprint graphs [23, 20]. A standard set of graph edit operations on an attribute graph includes insertion, and deletion and substitution of edges and nodes and substitution of vertex and edge labels. Unfortunately, there is no polynomial time algorithm for computing the exact graph edit distance between two graphs. However, several algorithms have been developed for computing approximated or sub-optimal graph edit distance in polynomial time [26, 29, 1, 9, 23]. A common strategy used for computing the GED is to find an optimal assignment between each node of one graph to each node of the other graph with minimum cost. The optimal assignment is computed by solving an assignment problem with a cost matrix derived using the structure of the graphs and the costs of graph edit operations. Graph edit distance has many applications in social network graph computation, pattern recognition and biometrics such as in fingerprint identification systems [23, 20].

Our Contributions. In this paper, for the first time, we consider secure two-party graph edit distance computation where each party has a private graph and they wish to jointly compute an approximated graph edit distance between two private graphs, without leaking any information about their input graph. A private graph is meant by the structure of the graph represented by an adjacency matrix, node labels and edge labels are private, only the number of nodes is public. First, we propose a general framework for securely computing approximated graph edit distance, which consists of securely computing the entries of the cost matrix from the private input graphs, securely solving the assignment problem and securely processing an optional phase to obtain the graph edit distance. Then, as an example, we develop a protocol for securely computing an approximated graph edit distance, determining the error-tolerant graph matching, based on the algorithm by Riesen and Bunke [26]. Our protocol construction relies on threshold additive homomorphic encryption scheme [13] instantiated by the threshold Paillier encryption scheme [25]. The reason for choosing homomorphic encryption in the construction is to design efficient protocols by exploiting the structures of the GED algorithms. To construct the main protocol, we develop several sub-protocols such as a private maximum computation protocol and an optimal assignment protocol based on the Hungarian algorithm. We prove the security of the protocol in the semi-honest model. The difference between the workloads of the parties is negligible. The asymptotic complexity for the proposed protocol is $O(n^5(\ell \log^*(\ell)))$, where ℓ is the bit length of ring elements and n is the maximum among the numbers of nodes in two graphs.

2 Related Work

2.1 Secure Two-party Computation

Secure two-party computation is a powerful tool that enables two parties to jointly compute a function on their private inputs without revealing any information about the inputs except the output of the function. Works on secure two-party computation began with the seminal work of Yao [28] that showed that any function can be securely evaluated in the presence of semi-honest adversaries by first generating a garbled circuit computing that function and then sending it to the other party. Then the other party can obtain the output by evaluating the garbled circuit using a 1-out-of-2 Oblivious Transfer (OT) protocol. A series of work on secure two-party computation have been done under different security settings and on optimization of garbled circuits [17, 4, 15], to name a few and a number of tools and compilers such as Fairplay [19] and TASTY [14] have been developed for secure computation.

2.2 Secure Processing of Graph Algorithms

Graph algorithms have a wide variety of use in many secure applications. Recently secure and data oblivious graph algorithms have been studied in [2, 5, 6]. Aly et al. [2] proposed secure data-oblivious algorithms for shortest path and maximum flow algorithms. In [6], Blanton et al. proposed secure data-oblivious algorithms for breadth-first search, single-source single-destination shortest path, minimum spanning tree, and maximum flow problems. In [5], Blanton and Saraph proposed secure data-oblivious algorithms for finding maximum matching size in a bipartite graph. In our work, as a sub-task, we need to find a perfect matching for computing the optimal cost in a complete weighted bipartite graph.

2.3 Secure Edit Distance Computation

An edit distance measures the dissimilarity (or similarity) between two strings. In [3], Atallah et al. proposed a privacy-preserving protocol for computing an edit distance between two strings based on an additive homomorphic encryption scheme. Jha et al. [16] presented privacy-preserving protocols for computing edit distance between two strings. The protocols are constructed using oblivious transfer and Yao's garbled circuits method. Later on, Huang et al. [15] developed a faster protocol for edit distance computation with the garbled circuit approach. Recently, Cheon et al. [7] proposed a privacy-preserving scheme for computing edit distance over encrypted strings. Their protocol is based on a somewhat homomorphic encryption scheme.

3 Preliminaries

In our construction, we use the threshold Paillier encryption scheme (TPS) $\mathsf{TPS} = (\pi_{\text{DistKeyGen}}, \pi_{\text{DistSk}}, \text{Enc}, \pi_{\text{DistDec}})$ in the two-party setting, due to Hazay el al. [13] where $\pi_{\text{DistKeyGen}}$ is the protocol for distributively generating a RSA modulus $N = pq$, π_{DistSk} is the protocol for distributed generation of shared private key and π_{DistDec} is the protocol for the distributed Paillier decryption of shared private key. The encryption algorithm Enc is defined as follows. For a plaintext message m with randomness $r \in_R \mathbb{Z}_N$ the ciphertext is computed as $c = \text{Enc}(m, r) = r^N(N + 1)^m \bmod N^2$. where $N = pq$ and p and q are two large primes of equal length. Assume that the bit length of N is ℓ. The Paillier encryption scheme has 1) additive homomorphic property: $E(m_1 + m_2) = \text{Enc}(m_1) \cdot \text{Enc}(m_2)$ and $\text{Enc}(km_1) = \text{Enc}(m_1)^k$ and 2) rerandomizing property meaning for a ciphertext c, without knowing the private key, another ciphertext $c' = \mathsf{Rand}(pk, \text{Enc}(m; r), r') = r'^N r^N (N + 1)^m = (rr')^N (N + 1)^m$ can be created. For the details about other protocols, the reader is referred to [13].

The computation of the GED involves operations on negative numbers as well. We represent the negative numbers in modular arithmetic in the encryption as $[\lceil \frac{N}{2} \rceil, N-1] \equiv [-\lfloor \frac{N}{2} \rfloor, -1]$. The positive numbers lie in the range $[0, \lfloor \frac{N}{2} \rfloor]$ and the negative numbers lie in the range $[\lceil \frac{N}{2} \rceil, N-1]$.

4 Problem Formulation

We consider an undirected attribute graph $G = (V, E, l_G, \zeta_G)$ where V is a finite set of vertices, E is the set of edges, and l_G is the vertex labeling function and ζ_G is the edge labeling function. Assume that the graph G does not contain any multi-edges and self-loops. Let $G_1 = (V_1, E_1, l_{G_1}, \zeta_{G_1})$ be a source graph and $G_2 = (V_2, E_2, l_{G_2}, \zeta_{G_2})$ be a target graph. The graph edit distance [1, 26] between G_1 and G_2 is defined by $f_{GED}(G_1, G_2) = \min_{(\mathsf{eo}_1,\ldots,\mathsf{eo}_k)\in\Gamma(G_1,G_2)} \sum_{i=1}^{k} c(\mathsf{eo}_i)$ where $\Gamma(G_1, G_2)$ is the set of all edit paths that transform G_1 into G_2 and $c(\mathsf{eo}_i)$ denotes the cost for the edit operation eo_i. The reader is referred to Appendix C for the details about graph edit operations.

In this work we consider a setting where there are two parties P_1 and P_2, P_1 has a private graph G_1 and P_2 has another private graph G_2. The parties wish to compute an approximated graph edit distance $f_{GED}(G_1, G_2)$ between G_1 and G_2 without leaking anything about their input graph, where f_{GED} is a function running in polynomial time computing an approximated graph edit distance between G_1 and G_2. At the end of the execution of the protocol, each party P_i should learn nothing about other party's input graph G_{3-i}, beyond the edit distance value $f_{GED}(G_1, G_2)$, $i = 1, 2$. A private graph is meant by node and edge labels and the structure of the graph represented by an adjacency matrix are private, only the number of nodes in the graph is public.

Adversary model. We define the security of the protocol for the GED computation against *honest-but-curious* or *semi-honest* adversaries where a party compromised by an adversary follows the prescribed actions of the protocol and aims to learn some unintended information from the execution of the protocol. Let $\mathcal{A}$ be a probabilistic polynomial time adversary that can corrupt at most one party at the beginning of the execution of the protocol. The adversary $\mathcal{A}$ sends all input messages of the corrupted party during the execution of the protocol and receives messages from the honest party. The honest party follows the instruction of the protocol.

Let $\mathcal{A}$ corrupts the party P_i. We denote the view of P_i in the real execution of the protocol Π by $\mathsf{VIEW}^{\Pi}_{P_i}(1^\lambda, G_1, G_2) = \{G_i, R_i, m_1, m_2, \cdots, m_T\}$, $i = 1$ or 2, where G_i is P_i's private input graph, $m_1, m_2, \cdots, m_T$ are the messages received from P_{3-i} and R_i is P_i's random tape used during the execution of the protocol.

Definition 1. *Let $f_{GED}(G_1, G_2)$ be the functionality computing an approximated graph edit distance. We say that a two-party protocol Π securely evaluates $f_{GED}(G_1, G_2)$ in the presence of semi-honest adversaries if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_{P_1}, \mathcal{S}_{P_2})$ such that for all G_1 and G_2, it holds that*

$$\{\mathcal{S}_{P_i}(1^\lambda, G_i, f_{GED}(G_1, G_2))\} \overset{c}{\approx} \{\mathit{VIEW}^{\Pi}_{P_i}(1^\lambda, G_1, G_2)\}$$

where $\overset{c}{\approx}$ denotes the computational indistinguishably of two distribution ensembles.

5 Description of Proposed GED Protocols

This section presents a framework for the two-party graph edit distance computation based on the assignment problem. As an example, we present a protocol for the graph edit distance computation and prove its security in the semi-honest model.

5.1 A Framework for Two-party GED Computation

Fig. 1 provides the process of an approximated GED computation. At a high level, the graph edit distance computation consists of three phases, namely the construction of the cost matrix, solving the optimal assignment problem with the cost matrix and further processing (optional processing) using the results from the assignment problem and inputs graphs to improve the approximated GED. The cost matrix construction phase takes graph inputs from the parties and computes the entries of the matrix in terms of the costs of graph edit operations. Solving the assignment problem does not take any graph inputs from parties. Based on the approximation factor of the approximated GED, the optional processing is performed. The general structure of the protocols for two-party graph edit distance computation consists of secure two-party evaluations of the cost matrix construction, the optimal assignment problem and optional processing.

At the end of secure processing of each phase, we ensure that there is no leakage of information from the output, except the final output that will be known to both parties.

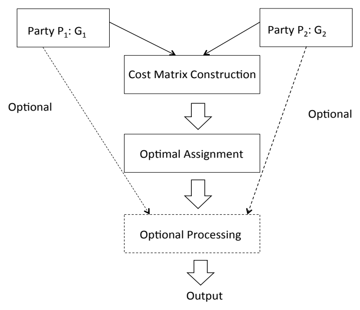

Figure 1: A block diagram for two-party graph edit distance computation

In the current paper, we perform the secure evaluation of graph edit distance, following the above framework, using the threshold Paillier additive homomorphic encryption scheme. The private key of the encryption scheme is shared between two parties. First, the parties construct an encrypted cost matrix using the input graphs and then they run the optimal assignment protocol on the encrypted cost matrix. The encrypted outputs from the optimal assignment protocol along with the input graphs if needed are used in the optional processing phase to obtain the graph edit distance. In Section 5.3, we present an approximated graph edit distance computation protocol.

5.2 Sub-Protocols

Secure equality testing and comparison protocols have been extensively studied in the literature under different two-party computation settings, e.g., in [8, 11, 27, 18]. We present a variant of encrypted equality test protocol, denoted by π_{EQ} in the Appendix. We use the greater-than protocol of Toft [27] with the modification that we replace the equality test protocol by π_{EQ}. In this section we present two sub-protocols `Private Maximum Computation` protocol and `Optimal Assignment` protocol that are necessary for the main protocols for graph edit distance. As our protocol construction uses an equality check, comparison, oblivious transfer and oblivious polynomial evaluation protocol, we denote the functionalities by $\mathcal{F}_{\text{EQ}}$, $\mathcal{F}_{\text{CMP}}$ $\mathcal{F}_{\text{OT}}$, and $\mathcal{F}_{\text{OPE}}$ and corresponding protocols by π_{EQ}, π_{CMP}, π_{OT} and π_{OPE}, respectively.

5.2.1 Private Maximum Computation Protocol

Let P_1 and P_2 hold a vector of encrypted numbers $\mathbf{c} = (c_1, c_2, ..., c_n)$ with $c_i = \text{Enc}(x_i)$ for the plaintext vector $\mathbf{x} = (x_1, x_2, ..., x_n)$. Let x_{mi} be the maximum value in $\mathbf{x}$ for index mi, $1 \leq \mathsf{mi} \leq n$. The private maximum computation (PMC) protocol is to jointly compute the encrypted maximum value $\text{Enc}(x_{\mathsf{mi}})$ and the encrypted index $\text{Enc}(\mathsf{mi})$ from $\mathbf{c}$ without revealing x_{mi} and mi.

We develop a two-party protocol for private maximum computation. The basic idea behind the construction of the PMC protocol is that one party shuffles the order of the elements of $\mathbf{c}$ through a secret permutation π_1 and after shuffling, each element is re-randomized using $\texttt{Rand}(\cdot, \cdot,)$. We denote the resultant vector by $\mathbf{c}'$. Next, the other party chooses a random permutation π_2 and using this permutation, it obliviously picks up an element from $\mathbf{c}'$ by running a 1-out-of-n oblivious transfer (OT) protocol [22], denoted by OT_1^n, and then randomizes the chosen element. Both parties then run a comparison protocol to determine the maximum value. This procedure is repeated $(n-1)$ times for $\pi_2(i), 2 \le i \le n$ to compute the maximum among n encrypted elements. The encrypted index Enc(mi) for the maximum value is computed through an oblivious polynomial evaluation (OPE) protocol. We use the FNP oblivious polynomial evaluation protocol [10] to obtain the encrypted index Enc(mi). We describe the details of the protocol in Fig. 2.

Protocol: `Private MAX Computation` π_{PMC}

Input: A ciphertext vector $\mathbf{c} = (c_1, c_2, ..., c_n)$ of $\mathbf{x} = (x_1, x_2, ..., x_n)$ where $c_i = \text{Enc}(x_i), 1 \le i \le n$.

Output: Encryption of the maximum value $\text{Enc}(x_{\mathsf{mi}})$ and its encrypted position $\text{Enc}(\mathsf{mi})$.

1. P_1 chooses a random permutation π_1 on $\{1, 2, ..., n\}$ and computes $(c_{\pi_1(1)}, c_{\pi_1(2)}, ..., c_{\pi_1(n)})$. It then randomizes this vector and obtains $\mathbf{c}' = (c'_1, c'_2, ..., c'_n)$ where $c'_i = \mathtt{Rand}(pk, c_{\pi_1(i)}, r_i), 1 \le i \le n$ where r_i is a random number.
2. P_2 chooses a random and secret permutation π_2 on $\{1, 2, ..., n\}$. It then runs an OT_1^n protocol with inputs $\mathbf{c}'$ from P_1 and $\pi_2(1)$ from P_2. Let $c'_{\pi_2(1)}$ be the output of the OT protocol. P_2 randomizes $c'_{\pi_2(1)}$ as $c''_1 = \mathtt{Rand}(pk, c'_{\pi_2(1)}, r'_1)$ and sends c''_1 to P_1.
3. Both parties set $c_{Index} \leftarrow c''_1$. P_2 assigns $Index \leftarrow \pi_2(1)$.
4. For each $t \in [2, n]$, P_1 and P_2 performs the following steps:
   (a) P_2 chooses $\pi_2(t)$.
   (b) P_1 and P_2 run the OT_1^n protocol with inputs $\mathbf{c}'$ from P_1 and $\pi_2(t)$ from P_2 Let $c'_{\pi_2(t)}$ be the output of the OT protocol received by P_2.
   (c) P_2 randomizes $c'_{\pi_2(t)}$ as $c''_t = \mathtt{Rand}(pk, c'_{\pi_2(1)}, r'_t)$ and sends c''_t to P_1.
   (d) P_1 and P_2 run the comparison protocol π_{CMP} with inputs c''_t and c_{Index} and let $\text{Enc}(b_{t-1})$ be the output. They run the threshold decryption protocol $\text{DistDec}(\text{Enc}(b_{t-1}))$. If $b_{t-1} = 1$, both parties update $c_{Index} \leftarrow c''_t$ and P_2 updates $Index \leftarrow \pi_2(t)$.
5. P_1 computes the polynomial representation of π_1^{-1} using the Lagrange interpolation with coefficients in $\mathbb{Z}_n$ and let $Q_{\pi_1^{-1}}(x) = \sum_{j=0}^{n-1} Q_j x^j$ be the polynomial of degree at most $(n-1)$.
6. P_1 and P_2 run the FNP OPE protocol with inputs $Q_{\pi^{-1}}(x)$ from P_1 and $Index$ from P_2 to compute the encrypted index Enc(mi) where $\mathsf{mi} = Q_{\pi_1^{-1}}(Index)$.
   (a) P_1 encrypts the coefficients of $Q_{\pi_1^{-1}}(x)$ as $(\text{Enc}(Q_0), \text{Enc}(Q_1), \cdots, \text{Enc}(Q_{n-1}))$ and sends it to P_2.
   (b) P_2 computes $\text{Enc}(Q_{\pi_1^{-1}}(Index)) = \prod_{j=0}^{n-1} (\text{Enc}(Q_1))^{Index^j}$ and sends $\text{Enc}(Q_{\pi_1^{-1}}(Index))$ to P_1.

Figure 2: Protocol for private maximum computation

Complexity. We evaluate the communication and computation overhead of the π_{PMC} protocol, which is composed of π_{CMP}, an OT_1^n protocol and an OPE protocol. Since the round complexity of π_{CMP} is $O(\log(\ell)\log^*(\ell))$, the total communication complexity for π_{CMP} is $(n\log(\ell)\log^*(\ell))$. The communication overhead for OT_1^n is $O(n)$. Therefore the overall communication complexity for π_{PMC} is $O(n^2 + n\ell\log(\ell)\log^*(\ell))$. It is easy to see that the computation complexity of the protocol is also $O(n^2 + n\ell\log(\ell)\log^*(\ell))$.

Theorem 1. *The protocol π_{PMC} securely computes the encrypted maximum value and its encrypted maximum index, in the presence of semi-honest adversaries.*

Proof. To prove π_{PMC} is secure in the presence of semi-honest adversaries, we need to show that π_{PMC} is secure in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{CMP}}, \mathcal{F}_{\mathrm{DistDec}}, \mathcal{F}_{\mathrm{OPE}})$-hybrid model. Observe that both parties receive the result from the protocol. Since we considered the semi-honest model, the adversaries are not allowed to alter their inputs. We construct two separate simulators for two parties and denote the simulator for P_1 by $\mathcal{S}_1$ and the simulator for P_2 by $\mathcal{S}_2$. Both simulators have independent uniform random tapes.

When P_1 is corrupted. Let $\mathcal{A}$ be an adversary controlling the party P_1. The simulator $\mathcal{S}_1$ receives security parameter 1^λ, the input $\mathbf{c}$ and outputs $\mathrm{Enc}(x_{\mathsf{mi}})$ and $\mathrm{Enc}(\mathsf{mi})$. In the real execution of the protocol, the parties's input is a common encrypted vector. The parties invoke the protocols for the functionalities $\mathcal{F}_{\mathrm{OT}}$, $\mathcal{F}_{\mathrm{CMP}}$ and $\mathcal{F}_{\mathrm{DistDec}}$ and $\mathcal{F}_{\mathsf{OPE}}$ during the execution of the protocol. The view of the party P_1 is given by

$$\mathsf{VIEW}_{P_1} = \big(\mathbf{c}, (\mathrm{Enc}(x_{\mathsf{mi}}), \mathrm{Enc}(\mathsf{mi})), \pi_1, \{r_i\}_{i=1}^n, \{c_i''\}_{i=1}^n, \{b_{t-1}\}_{t=1}^{n-1}, \mathrm{Enc}(Q_{\pi_1^{-1}}(Index))\big).$$

In the ideal execution, to compute $\mathrm{Enc}(x_{\mathsf{mi}})$, the parties run the $\mathcal{F}_{\mathrm{OT}}$, $\mathcal{F}_{\mathrm{CMP}}$ and $\mathcal{F}_{\mathrm{DistDec}}$ functionalities. In the protocol, the parties receive messages from the other party as well as the trusted third party computing the above functionalities. The simulators simulate the outputs of the trusted third party functionalities.

$\mathcal{S}_1$ chooses a permutation π_1 at uniformly random from $\mathsf{Sym}(n)$ denoting a symmetric group on n symbols and generates random tapes $\mathbf{r}_0 = \{r_i^0\}_{i=1}^n$ and computes $\mathbf{c}'$. $\mathcal{S}_1$ calls the trusted party computing $\mathcal{F}_{\mathrm{OT}}$ with inputs $\pi_2(1)$ and $\mathbf{c}'$. As an output, it receives noting. $\mathcal{S}_1$ receives c_1'' from P_2. The simulator performs the following steps for $t = 2, ..., n$. $\mathcal{S}_1$ calls the trusted party computing $\mathcal{F}_{\mathrm{OT}}$ with inputs $\pi_2(t)$ and $\mathbf{c}'$. As an output, it receives noting. $\mathcal{S}_1$ receives c_t'' from P_2. $\mathcal{S}_1$ randomly chooses a bit b_{t-1}' and a random tape for the encryption of $\mathrm{Enc}(b_{t-1}')$, which is a simulated value for the functionality $\mathcal{F}_{\mathrm{CMP}}$. $\mathcal{S}_1$ then calls the trusted third party $\mathcal{F}_{\mathrm{DistDec}}$ with the input $\mathrm{Enc}(b_{t-1}')$ and receives bit b_t. The simulator performs local computation for c_{Index} using the bit b_t. The simulator randomly chooses the coefficients $(Q_0, Q_1, \cdots, Q_n)$ of $Q(x)$. $\mathcal{S}_1$ simulates the output $\mathrm{Enc}(Q_{\pi_1^{-1}}(Index)$ of the trusted $\mathcal{F}_{\mathsf{OPE}}$ functionality with inputs $Q(x)$ and $Index$.

The view for the adversary $\mathcal{A}$ output by the simulator is $\mathcal{S}_1(\mathbf{c}, \mathrm{Enc}(x_{\mathsf{mi}}), \mathrm{Enc}(\mathsf{mi})) = (\mathbf{c}, \mathrm{Enc}(x_{\mathsf{mi}}), \mathrm{Enc}(\mathsf{mi}), \pi_1, \mathbf{r}_0, \mathbf{c}', \{c_t''\}_{i=1}^n, \{b_{t-1}'\}, \mathrm{Enc}(Q_{\pi_1^{-1}}(Index))$. The simulator does not have knowledge of the actual permutation π_2 in the real execution of the protocol. The view in the real execution of the protocol VIEW_{P_1} and the view of the simulator $\mathcal{S}_1$ are identically distributed since the Paillier encryption scheme is semantically secure and $\mathbf{r}_0$ is uniformly distributed.

When P_2 is corrupted. Let $\mathcal{A}$ be an adversary controlling the party P_2. The construction of the simulator $\mathcal{S}_2$ for the view of $\mathcal{A}$ is similar to that of $\mathcal{S}_1$, except $\mathcal{S}_2$ simulates the output for the trusted functionality $\mathcal{F}_{\mathrm{OT}}$ and it does not choose the coefficients of $Q(x)$. The view of the adversary output by $\mathcal{S}_2$ is given by $\mathcal{S}_2(\mathbf{c}, \mathrm{Enc}(x_{\mathsf{mi}}), \mathrm{Enc}(\mathsf{mi})) = (\mathbf{c}, \mathrm{Enc}(x_{\mathsf{mi}}), \mathrm{Enc}(\mathsf{mi}), \pi_2, \mathbf{r}_1, \{c_i'\}, \{b_{t-1})\})$.

In the real execution of the protocol, the actual permutation π_1 is unknown to the simulator. Applying the same arguments as above, the views for the adversary in the real execution of the protocol and ideal execution of the protocol are identically distributed. $\square$

5.2.2 Optimal Assignment (OA) Protocol

The assignment problem is one of the fundamental optimization problems. Given two sets $X = \{u_1, u_2, \cdots, u_n\}$ and $Y = \{v_1, v_2, \cdots, v_n\}$ and a cost matrix $W = (w_{ij})_{n\times n}$ where w_{ij} is the cost of assigning u_i to v_j, the assignment problem is to find a permutation ρ on $[1, n]$ that maximizes $\sum_{i=1}^{n} w_{i\rho(i)}$. We denote an assignment problem instance and its solution by $(\rho, \sum_{i=1}^{n} w_{i\rho(i)}) \leftarrow \mathsf{AssignProb}(X, Y, W)$, which can be solved by the Hungarian algorithm with time complexity $O(n^3)$ [21]. The assignment problem can also be viewed as the problem of finding a perfect bipartite matching in a complete weighted bipartite graph $G = (V, E, W)$ with $V = X \cup Y, X \cap Y = \phi$ where the cost matrix W is the weight matrix consisting of weights of the edges. In this paper, we consider the perfect bipartite matching variant of the Hungarian algorithm. An optimal assignment ρ that minimizes $\sum_{i=1}^{n} w_{i\rho(i)}$ can be obtained from this by making the entries of the cost matrix W negative.

Given an encrypted cost matrix $W = (\mathrm{Enc}(w_{ij}))_{n\times n}$ for $\mathsf{AssignProb}$(X, Y, W), we develop a two-party protocol for the assignment protocol based on the Hungarian algorithm for computing $\mathrm{Enc}(\sum_{i=1}^{n} w_{i\rho(i)})$ for an optimal assignment ρ. In the secure two-party computation protocol, we resolve the following challenges a) securely computing and updating the labeling of nodes in X and Y; b) hiding the edges in the perfect matching set as it eventually determines the optimal assignment ρ; and c) securely computing augmenting paths and updating the matching set. Since the order of node and/or edge operations during the execution of the algorithm leaks information about the assignment, we prevent this by encrypting the matching set $\mathcal{M}$ and shuffling the order of nodes while keeping the assignment problem invariant. We make the following observation about the assignment problem when it solved using the Hungarian algorithm.

Observation 1. *Let $(\rho, \sum_{i=1}^{n} w_{i\rho(i)}) \leftarrow \mathsf{AssignProb}(X, Y, W)$ be an assignment problem as described above. Let π be a permutation on $[1, n]$. Define $X^{\pi} = \{u_{\pi(1)}, \cdots, u_{\pi(n)}\}$ and $Y^{\pi} = \{v_{\pi(1)}, \cdots, v_{\pi(n)}\}$ and $W^{\pi} = (w_{\pi(i)\pi(j)})_{n\times n}$. If the assignment problems (X, Y, W) has an optimal value $\sum_{i=1}^{n} w_{i\rho(i)}$ with assignment mapping ρ, then the assignment problem $\mathsf{AssignProb}(X^{\pi}, Y^{\pi}, W^{\pi})$ has the same optimal value with assignment mapping $\rho_1 = \pi \circ \rho \circ \pi^{-1}$.*

Our main idea for constructing the OA protocol is to choose a secret permutation π shared between two parties and transform the problem $\mathsf{AssignProb}(X, Y, W)$ into $\mathsf{AssignProb}(X^{\pi}, Y^{\pi}, W^{\pi})$ and then securely execute the steps of the bipartite matching algorithm on the encrypted cost matrix. The party P_1 chooses a secret permutation π_1 and P_2 chooses another secret permutation π_2. Then they jointly construct the encrypted cost matrix $W^{\pi} = (\mathrm{Enc}(w_{\pi(i)\pi(j)}))$ where $\pi = \pi_2 \circ \pi_1$. We compute the initial labelings of nodes in X using the private maximum computation protocol π_{PMC}. We encrypt node identities $u_i \in X$ and $v_j \in Y$ of the bipartite graph and their labels, denoted by $\mathtt{lbl}_X(u)$ for $u \in X$ and $\mathtt{lbl}_Y(v)$ for $v \in Y$ and construct 2-tuple sequences as $(\mathrm{Enc}(u_i), \mathrm{Enc}(\mathtt{lbl}_X(u_i))), 1 \le i \le n$ for both X and Y. We use the same permutation π to hide the order of each sequence of 2-tuple encrypted values component-wise for both X and Y. Denoting $\mathcal{M} = \{(\mathrm{Enc}(u), \mathrm{Enc}(v)) : u \in X, v \in Y\}$ by the matching set containing encrypted edges, $\{\mathrm{Enc}(u) : u \in X\}$ the set all encrypted tail node ids in $\mathcal{M}$ by

$\mathcal{M} \star X$ and $\{\text{Enc}(v) : v \in Y\}$ the set all encrypted head node ids in $\mathcal{M}$ by $\mathcal{M} \star Y$. The initial matching is found by using the π_{EQ} and π_{DistDec} protocol. An encrypted equality graph $EQ^{\mathtt{lbl}}$ represented by an encrypted adjacency matrix is constructed from encrypted labels for X and Y and encrypted cost matrix W using the π_{EQ} protocol. The perfect matching is found by extending the matching set by finding an encrypted augmenting path. An encrypted augmenting path is found by executing the breadth-first-search (BFS) algorithm on the encrypted equality graph $EQ^{\mathtt{lbl}}$ where the source and target vertices are free vertices in X and Y. We adopt a variant of Blanton et al.'s BFS algorithm [6] in our setting where the secret key for the decryption algorithm is shared between two parties and we denote this protocol by π_{BFS}. We don't provide the technical details due to space limit. For an encrypted equality graph $\mathcal{P} := \text{Enc}(t_0) - \text{Enc}(t_1) - \text{Enc}(t_2) - \cdots - \text{Enc}(t_k)$ of length $k-1$, the set of encrypted edges are given by $\mathcal{P}_{edge} = \{(\text{Enc}(t_0), \text{Enc}(t_1)), (\text{Enc}(t_2), \text{Enc}(t_1)), \cdots, (\text{Enc}(t_{k-1}), \text{Enc}(t_k))\}$. After finding $\mathcal{P}_{edge}$, the matching set is updated as $M \leftarrow M \Delta \mathcal{P}_{edge}$ where Δ is the symmetric difference set operation. We use two dummy counters of length n for keeping track of encrypted free nodes of X and Y. For computing the GED, we only need the maximum value $\sum_{i=1}^{n} w_{i\rho(i)}$. Thus the protocol outputs only $\sum_{i=1}^{n} w_{i\rho(i)}$. Fig. 3 presents the details of our secure protocol for the assignment problem.

Complexity. From π_{OA}, it can be seen that the time complexity for finding the initial matching (Step 1 to Step 7) is $O(n^3 + n^2\ell \log(\ell) \log^*(\ell))$. If the initial matching is not a perfect matching, the computational complexity for terminating the protocol is $O(n^5\ell \log^*(\ell) + n^4\ell \log(\ell) \log^*(\ell)) = O(n^5\ell \log^*(\ell))$. An insecure version of the Hungarian algorithm runs in $O(n^3)$ steps. The overhead of the protocol due to security is $(n^2\ell \log^*(\ell))$.

Theorem 2. *The protocol π_{OA} securely computes the encrypted optimal value in the presence of semi-honest adversaries.*

Proof. We prove the security of the protocol in the hybrid model. In the protocol, one party receives messages from the other party and also from the trusted third party computing a functionality. The simulator also needs to simulate the outputs for the trusted third party functionalities. We construct two different simulators for the view of the adversary.

When P_1 is corrupted. Let $\mathcal{A}$ be the adversary controlling the party P_1. $\mathcal{S}_1$ chooses $2n$ uniformly random tapes $\mathbf{r}_0 = \{(r_i^0, r_i^1)\}_{i=1}^{n}$ from $\mathbb{Z}_N$ and computes the encrypted 2-tuple vector LBL^Y. It emulates the outputs $\text{Enc}(\mathsf{lbl}_X(u_i))$ and $\text{Enc}(d_i), 1 \le i \le n$ for the trusted third party functionality $\mathcal{F}_{\mathsf{PMC}}$ on ith row W_i. $\mathcal{S}_1$ chooses n random tapes $\mathbf{r}_1 = \{r_i^2\}_{i=1}^{n}$ uniformly at random for P_1 and computes the encryptions of $i, 1 \le i \le n$. $\mathcal{S}_1$ picks a permutation π_1 and $(6n + n^2)$ random tapes $\mathbf{r}_2 = \{\{(r_i^3, r_i^4, r_i^5, r_i^6, r_i^7, r_i^8)\}_{i=1}^{n}, (r_{ij}^0)_{n\times n}\}$ uniformly at random for P_1 and computes $LBL^{X_{\pi_1}}$, $LBL^{Y_{\pi_1}}$, D^{π_1} and W^{π_1} and rerandomizes each encrypted value. $\mathcal{S}_1$ chooses π_2 and $(6n + n^2)$ random tapes $\mathbf{r}_3 = \{\{(r_i^3, r_i^4, r_i^5, r_i^6, r_i^7, r_i^8)\}_{i=1}^{n}, (r_{ij}^1)_{n\times n}\}$ uniformly at random and computes $LBL^{X_{\pi_2\circ\pi_1}}$, $LBL^{Y_{\pi_2\circ\pi_1}}$, $D^{\pi_2\circ\pi_1}$ and $W^{\pi_2\circ\pi_1}$ and rerandomizes each encrypted value using $\mathbf{r}_3$. In Step 5, for each $\text{Enc}(d_{\pi(i)}), 1 \le i \le n$, the simulator generates b_{ij} at random and computes $\text{Enc}(b_{ij})$ for $\mathcal{F}_{\mathsf{EQ}}$ with inputs $\text{Enc}(d_{\pi(i)})$ and $\text{Enc}(d_{\pi(j)}), 1 \le j \le i-1$ and obtains $\mathbf{b}_1 = (b_1, b_2, \cdots, b_n)$. $\mathcal{S}_1$ computes R from $\mathbf{b}_1$. $\mathcal{S}_1$ computes $\mathcal{M}$. In Steps 10 - 14, $\mathcal{S}_1$ simulates the output of the functionalities $\mathcal{F}_{\mathsf{EQ}}$, $\mathcal{F}_{\mathsf{DistDec}}$, $\mathcal{F}_{\mathsf{PMC}}$ and $\mathcal{F}_{\mathsf{BFS}}$ while ensuring the loop terminates in $O(n^3)$ steps. The outputs at ℓ-th iteration for Steps 10 - 14 are $\mathbf{b}_3^{\ell} = (b_1, b_2, \cdots, b_n)$ (Step 10); $\mathbf{z}^{\ell} = (z_1, z_2, \cdots, z_n)$ and $\mathbf{b}_4^{\ell} = (b_1, b_2, \cdots, b_n)$ (Step 11); $\mathbf{b}_5^{\ell} = (b_1, b_2, \cdots, b_{|T|\cdot|N_{\mathtt{lbl}}(S)|})$ (Step

Protocol: $\texttt{Optimal Assignment}$ based on the Hungarian algorithm π_{OA}
Input: The cost matrix $\text{Enc}(W) = (\mathsf{Enc}(w_{ij}))_{n\times n}$ $w_{ij} = cost(u_i, v_j)$.
Output: Optimal assignment value $\text{Enc}(\sum_{i=1}^{n} w_{i\rho(i)})$.

1. P_1 computes $(\text{Enc}(\texttt{lbl}_Y(v_1) : 1 \leq i \leq n)$ with $\texttt{lbl}_Y(v_i) = 0, v_i \in Y$ and $(\text{Enc}(i) : 1 \leq i \leq n)$ and constructs $LBL^Y = \Big((\text{Enc}(\texttt{lbl}_Y(v_i)), \text{Enc}(i)) : 1 \leq i \leq n\Big)$ and sends it to P_2.
2. $\mathcal{L} \leftarrow \phi$; $\mathcal{VP} \leftarrow \phi$; $\mathcal{M} \leftarrow \phi$;
3. For each $u_i \in X$, P_1 and P_2 run π_{PMC} with input ith row $W_i = (w_{i1}, w_{i2}, ..., w_{in})$ and obtain output $\text{Enc}(\texttt{lbl}_X(u_i))$ and $\text{Enc}(d_i)$ where $\texttt{lbl}_X(u_i) = w_{id_i} = \max\limits_{v_j \in Y}\{w_{ij}\}, u_i \in X$ and $1 \leq d_i \leq n$.
   (a) Construct $LBL^X = \Big((\text{Enc}(\texttt{lbl}_X(u_i)), \text{Enc}(i)) : 1 \leq i \leq n\Big)$.
   (b) Construct $D = ((\text{Enc}(i), \text{Enc}(d_i)) : 1 \leq i \leq n)$.
   (c) Update $\mathcal{L} \leftarrow \mathcal{L} \cup \{\big(\text{Enc}(\texttt{lbl}_X(u_i)), \text{Enc}(\texttt{lbl}_Y(v_i))\big)\}$.
4. P_1 chooses a random perm π_1 and computes the following and sends all to P_2
   (a) $LBL^{Y_{\pi_1}} := \Big((\text{Enc}(\texttt{lbl}_Y(v_{\pi_1(j)})), \text{Enc}(\pi_1(j))) : 1 \leq j \leq n\Big)$, $LBL^{X_{\pi_1}} := \Big((\text{Enc}(\texttt{lbl}_X(u_{\pi_1(j)})), \text{Enc}(\pi_1(j))) : 1 \leq j \leq n\Big)$
   (b) $D^{\pi_1} := ((\text{Enc}(\pi_1(j)), \text{Enc}(d_{\pi_1(j)})) : 1 \leq j \leq n)$
   (c) $W^{\pi_1} = (\text{Enc}(w_{\pi_1(i)\pi_1(j)}))_{n\times n}$
   (d) Rerandomize each encrypted value above
5. P_2 chooses a random perm π_2 and computes the following and sends all to P_1
   (a) $LBL^{Y_{\pi_2\circ\pi_1}} := \Big((\text{Enc}(\texttt{lbl}_Y(v_{\pi_2\circ\pi_1(j)})), \text{Enc}(\pi_2 \circ \pi_1(j))) : 1 \leq j \leq n\Big)$,
   (b) $LBL^{X_{\pi_2\circ\pi_1}} := \Big((\text{Enc}(\texttt{lbl}_X(u_{\pi_2\circ\pi_1(j)})), \text{Enc}(\pi_2 \circ \pi_1(j))) : 1 \leq j \leq n\Big)$
   (c) $D^{\pi_2\circ\pi_1} := ((\text{Enc}(\pi_2 \circ \pi_1(j)), \text{Enc}(d_{\pi_2\circ\pi_1(j)})) : 1 \leq j \leq n)$
   (d) $W^{\pi_2\circ\pi_1} = (\text{Enc}(w_{\pi_2\circ\pi_1(i)\pi_2\circ\pi_1(j)}))_{n\times n}$
   (e) Rerandomize each encrypted value above. Set $\pi = \pi_2 \circ \pi_1$.
6. For each $(\text{Enc}(\pi(i)), \text{Enc}(d_{\pi(i)})) \in D^{\pi}$, $i = 1, ..., n$, P_1 and P_2 run π_{EQ} protocol with inputs $\text{Enc}(d_{\pi(i)})$ and $\text{Enc}(d_{\pi(j)})$ and obtain $\text{Enc}(b_{ij}), b_{ij} \in \{0, 1\}$ for $j = 1, ..., i-1$. Compute $R = \prod_{j=1}^{i-1} \text{Enc}(b_{ij})$. P_1 and P_2 run π_{EQ} protocol with inputs R and $\text{Enc}(0)$ and obtain $\text{Enc}(b_i)$ as output. P_1 and P_2 then jointly decrypt $\text{Enc}(b_i)$. If $b_i = 1$, perform $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\text{Enc}(\pi(i)), \text{Enc}(d_{\pi(i)}))\}$.
7. If $|\mathcal{M}| = n$, **return** the encrypted optimal value is $\text{Enc}(\sum_{i=1}^{n} \texttt{lbl}_X(u_{\pi(i)}) + \sum_{i=1}^{n} \texttt{lbl}_Y(v_{\pi(i)})) = \prod_{i=1}^{n} \text{Enc}(\texttt{lbl}_X(u_{\pi(i)})) \prod_{i=1}^{n} \text{Enc}(\texttt{lbl}_Y(v_{\pi(i)})$. Else, P_1 and P_2 execute the following steps.
8. P_1 and P_2 construct a matrix $EQ^{\texttt{lbl}} = (\text{Enc}(e_{ij}))_{n\times n}$ by running the π_{EQ} protocol with inputs $\text{Enc}(\texttt{lbl}_X(u_{\pi(i)}) + \texttt{lbl}_Y(v_{\pi(j)}))$ and $\text{Enc}(w_{\pi(i)\pi(j)})$ where $\text{Enc}(e_{ij})$ is the output $1 \leq i, j \leq n$ and $e_{ij} \in \{0, 1\}$.

Protocol: `Optimal Assignment` π_{OA} (Cont.)

9. Initialize $S \leftarrow \phi$ and $T \leftarrow \phi$.
10. P_1 and P_2 find $\mathrm{Enc}(u_{\pi(i)})$ such that $\mathrm{Enc}(u_{\pi(i)}) \notin \mathcal{M} \star X$, then $S \leftarrow S \cup \{\mathrm{Enc}(u_{\pi(i)})\}$.
11. P_1 and P_2 compute $N_{\mathsf{lbl}}(S)$ for each $\mathrm{Enc}(u_{\pi(i)}) \in S$ as
    (a) For row $EQ_i^{\mathsf{lbl}} = (\mathrm{Enc}(e_{i1}), \mathrm{Enc}(e_{i2}), \cdots, \mathrm{Enc}(e_{in}))$ of $EQ^{\mathtt{lbl}}$, compute $Z_i = (\mathrm{Enc}(e_{i1} \cdot v_1), \mathrm{Enc}(e_{i2} \cdot v_2), \cdots, \mathrm{Enc}(e_{in} \cdot v_n))$ from $EQ_i^{\mathtt{lbl}}$ where $\mathrm{Enc}(e_{ik} \cdot k) = \mathrm{Enc}(e_{ik})^k$.
    (b) Run π_{SR} protocol with input $Z_i = (\mathrm{Enc}(e_{i1} \cdot v_1), \mathrm{Enc}(e_{i2} \cdot v_2), \cdots, \mathrm{Enc}(e_{in} \cdot v_n))$ and obtain the output $Z_i' = (z_1, z_2, \cdots, z_n)$
    (c) Run π_{EQ} with inputs z_j and $\mathrm{Enc}(0)$ and obtain the output $\mathrm{Enc}(b_j)$. Run DistDec on input $\mathrm{Enc}(b_i)$ and obtain b_j for $1 \leq j \leq n$. If $b_j = 0$, perform $N_{\mathtt{lbl}}(S) \leftarrow N_{\mathtt{lbl}}(S) \cup \{\mathrm{Enc}(v_j)\}$.
12. P_1 and P_2 check the equality of sets $N_{\mathtt{lbl}}(S)$ and T running π_{EQ} and π_{DistDec} protocols.
13. If $N_{\mathtt{lbl}}(S) = T$
    (a) P_1 and P_2 compute $\bar{T} = ((LBL_1^{Y_\pi} \star Y) - T)$ from sets $LBL^{Y_\pi} \star Y$ and T by running π_{EQ} and π_{DistDec} protocols.
    (b) For each $\mathrm{Enc}(u_{\pi(i)}) \in S$ and $\mathrm{Enc}(v_j) \in \bar{T}$, P_1 and P_2 compute $\mathrm{Enc}(lbl_{ij}) = \mathrm{Enc}(\mathtt{lbl}_X(u_{\pi(i)}) + \mathtt{lbl}_Y(v_j) - w_{\pi(i)\pi(j)})$.
    (c) P_1 and P_2 compute $\mathrm{Enc}(\delta_{\mathtt{lbl}}) = \min\{\mathrm{Enc}(lbl_{ij}) : \mathrm{Enc}(i) \in S, \mathrm{Enc}(j) \in T\}$ using the π_{PMC} protocol.
    (d) P_1 and P_2 update the label `lbl` as

$$\begin{aligned} \mathrm{Enc}(\mathtt{lbl}_X(u)) &= \mathrm{Enc}(\mathtt{lbl}_X(u)) \cdot \mathrm{Enc}(\delta_{\mathtt{lbl}})^{-1} && \text{if } E(u) \in S \\ \mathrm{Enc}(\mathtt{lbl}_Y(v)) &= \mathrm{Enc}(\mathtt{lbl}_Y(v)) \cdot \mathrm{Enc}(\delta_{\mathtt{lbl}}) && \text{if } E(v) \in T \end{aligned}$$

14. If $N_{\mathtt{lbl}}(S) \neq T$
    (a) P_1 and P_2 choose $\mathrm{Enc}(v_j) \in N_{\mathtt{lbl}}(S) - T$.
    (b) If $\mathrm{Enc}(v_j) \notin \mathcal{M} \star Y$, find an augmenting path $\mathcal{P} := \mathrm{Enc}(u_k) - \mathrm{Enc}(v_j)$ by running the π_{BFS} protocol with inputs $EQ^{\mathtt{lbl}}$ and $\mathrm{Enc}(v_j)$.
    (c) Update $\mathcal{M} \leftarrow \mathcal{M} \Delta \mathcal{P}_{edge}$. Goto Step 7.
    (d) If $\mathrm{Enc}(v_j) \in \mathcal{M} \star Y$ and $(\mathrm{Enc}(u_{\pi(\ell)}), \mathrm{Enc}(v_j)) \in \mathcal{M}$, extend alternating tree $S \leftarrow S \cup \{\mathrm{Enc}(u_{pi(\ell)})\}$ and $T \leftarrow T \cup \{\mathrm{Enc}(v_j)\}$. Goto Step 11.

Figure 3: Secure Optimal Assignment Protocol based on the Hungarian Algorithm

12); $\mathbf{b}_6^\ell = (b_1, b_2, \cdots, b_t), t \leq n$, (Step 13); $\mathbf{b}_7^\ell = (b_1, b_2, \cdots, b_t), t \leq n$, $\mathcal{P}_{sim}^\ell = \text{Enc}(u) - \text{Enc}(v)$ (simulated augmenting path), $\mathbf{b}_8^\ell = (b_1, b_2, \cdots, b_{|\mathcal{M}|\cdot|\mathcal{P}_{edge}|})$ and $\mathbf{b}_9^\ell = (b_1, b_2, \cdots, b_t), t \leq n$ (Step 14). Define $\mathbf{B}^\ell = (\mathbf{b}_3^\ell, \mathbf{z}^\ell, \mathbf{b}_5^\ell, \mathbf{b}_6^\ell, \mathbf{b}_7^\ell, \mathcal{P}_{sim}^\ell, \mathbf{b}_8^\ell, \mathbf{b}_9^\ell)$. The output of $\mathcal{S}_1$ is $\mathcal{S}_1(1^\lambda, n, X, Y, W) = (\mathbf{r}_1, \pi_1,\ \mathbf{r}_2, LBL^{X_{\pi_2\circ\pi_1}}, LBL^{Y_{\pi_2\circ\pi_1}}, D^{\pi_2\circ\pi_1}, W^{\pi_2\circ\pi_1}, EQ^{\mathbf{lbl}}, \mathbf{b}_1, \mathbf{b}_2, \{\mathbf{B}^\ell\})$. The distributions for $LBL^{X_{\pi_2\circ\pi_1}}, LBL^{Y_{\pi_2\circ\pi_1}}$, $D^{\pi_2\circ\pi_1}$ and $\mathbf{B}^\ell$ in the real and ideal executions are identically distributed since the random tapes for $\mathbf{r}_1$ and $\mathbf{r}_2$ were chosen uniformly at random, the Paillier encryption scheme is semantically secure and the permutation π_2 for the honest party in the real execution of the protocol is unknown to $\mathcal{S}_1$.

When P_2 is corrupted. Let the adversary $\mathcal{A}$ controlling the party P_2. The construction of the simulator is similar to that of $\mathcal{S}_1$, except Step 1. We don't provide the details of the simulator $\mathcal{S}_2$. The view of $\mathcal{A}$ output by $\mathcal{S}_2$ is $\mathcal{S}_2(1^\lambda, n, X, Y, W) = (\mathbf{r}_3, \pi_2, LBL^{X_{\pi_1}}, LBL^{Y_{\pi_1}}, D^{\pi_1}, W^{\pi_1}, EQ^{\mathbf{lbl}}, \mathbf{b}_1, \mathbf{b}_2, \{\mathbf{B}^\ell\})$. Applying the similar argument, the views for the adversary in the real and ideal execution of the protocol are identically distributed.

As the protocols π_{EQ}, π_{PMC}, π_{SR}, π_{DistDec} and π_{BFS} are secure, applying the composition theorem, π_{OA} is secure in the hybrid model against semi-honest adversaries and hence π_{OA} is secure in the real execution of the protocol. □

5.3 The Main Protocol for Graph Edit Distance

In this section we present a secure realization of the approximated GED computation by Riesen and Bunke [26], based on bipartite graph. We consider the two-party computation in the semi-honest model. We consider a setting where there are two parties P_1 and P_2, each party has a private graph $G_i = (V_i, E_i, l_{G_i}, \zeta_{G_i})$ with $n_i = |V_i| \geq 3$ and $n = n_1 + n_2$. For simplicity, we consider the cost matrix W that includes only the costs of node edit operations [1]. The parties start the protocol execution by computing a cost matrix. We start by explaining how the parties jointly construct the cost matrix.

5.3.1 Encrypted Cost Matrix Construction

We assume that each party secretly defines the costs for the graph edit operations deletion, insertion and substitution of nodes and/or edges. The edit operation costs for nodes are defined as follows. Let $l_{G_1}(u_i) = \alpha_i \in \mathbb{Z}_N$ be the node labeling function of G_1 and $l_{G_2}(v_j) = \beta_j \in \mathbb{Z}_N$ be the node labeling function of G_2. The party P_1 chooses the edit costs of node insertion and deletion operations as $c(u_i \to \epsilon) = c(\epsilon \to u_i) = C_1 \in \mathbb{Z}_N$. Similarly, the party P_2 decides the costs of insertion and deletion operations for nodes as $c(\epsilon \to v_j) = c(v_j \to \epsilon) = C_2 \in \mathbb{Z}_N$. The cost of the node substitution operation is defined as $w_{ij} = c(u_i \to v_j) = \min\{(c(u_i \to \epsilon) + c(\epsilon \to v_j)), c'(u_i \to v_j)\} = \min\{(C_1 + C_2), |\alpha_i - \beta_j|\}$ where $c'(u_i \to v_j) = |\alpha_i - \beta_j|$, $\alpha_i, \beta_j \in \mathbb{Z}_N$. This definition of the cost function can be found in [23]. Each entry of the cost matrix is computed by running a joint protocol.

We now explain how to construct the encrypted cost matrix $W = (\text{Enc}(w_{ij}))_{n\times n}$. For insertion and deletion operations, the party P_i encrypts its cost $\text{Enc}(C_i)$ and sends it to the other party. For the substitution cost, the parties exchange respective encrypted costs of insertion and deletion operations and encrypted node labels. Let the parties P_1 and P_2 have encryptions $\text{Enc}(d_1)$ and $\text{Enc}(d_2)$ of numbers d_1 and d_2, respectively and they would like to compute $\text{Enc}(|d_1-$

[1]Several constructions of cost matrix can be found in [26, 9, 24] for the improvement of the approximation of the actual GED. However, the two-party computation of GED remains same, except the cost matrix construction.

Protocol: `Protocol` Π_{GED}
Input: Graph $G_1 = (V_1, E_1, l_{G_1})$ from P_1 and $G_2 = (V_2, E_2, l_{G_2})$ from P_2.
Output: Approximated graph edit distance $d = f_{GED}(G_1, G_2)$ or $\perp$.

1. P_1 and P_2 run the distributed key generation protocol $\pi_{\mathsf{DistKeyGen}}$, followed by the distributed shared secret key generation protocol π_{DistSk}.
2. P_1 encrypts the cost for node insertion and deletion $\mathrm{Enc}(C_1)$ and $\{\mathrm{Enc}(\alpha_i) : u_i \in V_1\}$.
3. P_2 encrypts the cost for node insertion and deletion $\mathrm{Enc}(C_2)$ and $\{\mathrm{Enc}(\beta_j) : v_j \in V_2\}$.
4. For each $u_i \in V_1$ and $v_j \in V_2$, P_1 and P_2 run π_{Sub} with inputs $\mathrm{Enc}(C_1)$ and $\mathrm{Enc}(\alpha_i)$ from P_1 and inputs $\mathrm{Enc}(C_2)$ and $\mathrm{Enc}(\beta_j)$ from P_2 and obtain the output $\mathrm{Enc}(w_{ij})$.
5. P_1 and P_2 make all the entries of $E(W)$ negative, i.e., $W' = (\mathrm{Enc}(w'_{ij}))_{n\times n} = (\mathrm{Enc}(w_{ij})^{-1})_{n\times n}$ where $w'_{ij} = -w_{ij}$.
6. Run the optimal assignment protocol π_{OA} with input $\mathrm{Enc}(W')$ and obtain the output the encrypted the minimum cost $\mathrm{Enc}(\sum_{i=1}^{n} w'_{id_i})$.
7. P_1 and P_2 jointly run the distributed decryption protocol π_{DistDec} on $\mathrm{Enc}(\sum_{i=1}^{n} w'_{id_i})^{-1}$ and obtain the approximated graph edit distance $d = (\sum_{i=1}^{n} w_{id_i})$.

Figure 4: Protocol for computing an approximated graph edit distance based on bipartite graph

$d_2|)$ where $|d_1 - d_2|$ is the absolute difference between d_1 and d_2. The absolute difference between d_1 and d_2 can be computed as $|d_1 - d_2| = (d_1 - d_2) + b(d_2 - d_1) = (1-b)d_1 + (b-1)d_2$ where $b = 0$, $d_1 < d_2$; otherwise, $b = 1$. We use this relation to compute encrypted absolute difference between two encrypted numbers. We provide the details of the protocol in Figure 6 in Appendix A.

5.3.2 Description of the Protocol

We are now ready to describe the protocol. The parties P_1 and P_2 initiate the protocol by generating the public key and the shares of the private key for the threshold Paillier encryption scheme using $\pi_{\mathsf{DistKeyGen}}$ and π_{DistSk}, respectively. Each party encrypts its node labels for the construction of the encrypted cost matrix. The computation of GED consists of two main phases. First, the parties construct the encrypted cost matrix $\mathrm{Enc}(W) = (\mathrm{Enc}(w_{ij}))_{n\times n}$ using the function defined above and then solve the assignment problem with input as the encrypted cost matrix $\mathrm{Enc}(W)$ to find an optimal of the nodes of the graphs. The parties use the distributed decryption protocol DistDec to obtain the graph edit distance $f_{GED}(G_1, G_2) = (\sum_{i=1}^{n} w_{id_i})$. Fig. 4 presents the details of the approximated GED computation protocol.

Complexity of Π_{GED}. For computing encrypted node labels, each party performs $O(n)$ operations. The computation complexity for constructing the encrypted cost matrix is $O(n^2)$. The parties run the optimal assignment protocol on the encrypted matrix. The computational complexity and the communication complexity of the graph edit distance protocol Π_{GED} is at most $O(n^5 \ell \log^*(\ell))$. The complexity of the protocol is dominated by that of the optimal assignment protocol. In the protocol execution, the parties do almost an equal amount of computation.

Theorem 3. *Assuming the threshold Paillier encryption scheme is secure, the protocol* Π_{GED} *is secure in the presence of the semi-honest adversaries.*

Proof. The protocol Π_{GED} sequentially invokes the protocols for distributed key generation $\pi_{\mathsf{DistKeyGen}}$ and π_{DistSk}, the cost matrix construction π_{Sub} and the optimal assignment π_{OA}, and the distributed decryption π_{DistDec} for the approximated GDE. The protocols $\pi_{\mathsf{DistKeyGen}}$, π_{DistSk} and π_{DistDec} are secure according to [13]. The construction of π_{Sub} based on π_{CMP}. The security of the π_{Sub} protocol relies on that of π_{CMP}, which is proven secure in [27]. Theorem 2 guarantees the parties securely solves the assignment problem. According to the sequential composition theorem [12], Π_{GED} is secure against semi-honest adversaries in the real execution of the protocol. □

6 Conclusions

In this paper we considered secure two-party computation of graph edit distance measuring the dissimilarity between two graphs where each party has a private graph and they wish to jointly compute graph edit distance of two private graphs. We proposed a framework for the graph edit distance computation and, as an example, developed a protocol for computing of graph edit distance. To construct main protocols for graph edit distance, we developed sub-protocols such as private maximum computation and optimal assignment protocol based on the Hungarian algorithm. The asymptotic complexities of both protocols are $O(n^5(\ell \log^*(\ell)))$. Our protocol is secure against semi-honest adversaries and has applications in two-party social network graph computations for measuring structural similarity and fingerprint identifications.

Acknowledgement. This work was supported by ONR grant N00014-14-1-0029 and a grant from the King Abdulaziz City for Science and Technology (KACST). The authors would like to thank Seny Kamara for conducting several discussions during the initial phase of this work. The authors also thank the anonymous reviewers of CANS 2016 for bringing the references [3, 16] into our attention and for their helpful comments.

References

[1] Charu C. Aggarwal and Haixun Wang. *Managing and Mining Graph Data.* Springer US, 2010.

[2] Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Vyve. *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, chapter Securely Solving Simple Combinatorial Graph Problems, pages 239–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] Mikhail J. Atallah, Florian Kerschbaum, and Wenliang Du. Secure and private sequence comparisons. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, WPES '03, pages 39–44, New York, NY, USA, 2003. ACM.

[4] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 784–796, New York, NY, USA, 2012. ACM.

[5] Marina Blanton and Siddharth Saraph. *Computer Security – ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, chapter Oblivious Maximum Bipartite Matching Size Algorithm with Applications to Secure Fingerprint Identification, pages 384–406. Springer International Publishing, Cham, 2015.

[6] Marina Blanton, Aaron Steele, and Mehrdad Alisagari. Data-oblivious graph algorithms for secure computation and outsourcing. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 207–218, New York, NY, USA, 2013. ACM.

[7] Jung Hee Cheon, Miran Kim, and Kristin Lauter. *Homomorphic Computation of Edit Distance*, pages 194–212. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[8] Ivan Damgrd, Matthias Fitzi, Eike Kiltz, JesperBuus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304. Springer Berlin Heidelberg, 2006.

[9] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. Speeding up graph edit distance computation through fast bipartite matching. In Xiaoyi Jiang, Miquel Ferrer, and Andrea Torsello, editors, *Graph-Based Representations in Pattern Recognition*, volume 6658 of *Lecture Notes in Computer Science*, pages 102–111. Springer Berlin Heidelberg, 2011.

[10] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer Berlin Heidelberg, 2004.

[11] Craig Gentry, Shai Halevi, Charanjit S. Jutla, and Mariana Raykova. Private database access with he-over-oram architecture. *IACR Cryptology ePrint Archive*, 2014:345, 2014.

[12] Oded Goldreich. *Foundations of Cryptography Vol. II Basic Applications.* Cambridge University Press, 2004.

[13] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, and Tomas Toft. Efficient rsa key generation and threshold paillier in the two-party setting. In *Proceedings of the 12th Conference on Topics in Cryptology*, CT-RSA'12, pages 313–331, Berlin, Heidelberg, 2012. Springer-Verlag.

[14] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: Tool for automating secure two-party computations. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 451–462, New York, NY, USA, 2010. ACM.

[15] Yan Huang, David Evans, Jonathan Katz, and Lior 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.

[16] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. Towards practical privacy for genomic computation. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 216–230, Washington, DC, USA, 2008. IEEE Computer Society.

[17] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, EUROCRYPT '07, pages 52–78, Berlin, Heidelberg, 2007. Springer-Verlag.

[18] Helger Lipmaa and Tomas Toft. Secure equality and greater-than tests with sublinear online complexity. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 645–656, 2013.

[19] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[20] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[21] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[22] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 448–457, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[23] Michel Neuhaus and Horst Bunke. *Audio- and Video-Based Biometric Person Authentication: 5th International Conference, AVBPA 2005, Hilton Rye Town, NY, USA, July 20-22, 2005. Proceedings*, chapter A Graph Matching Based Approach to Fingerprint Classification Using Directional Variance, pages 191–200. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[24] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In Dit-Yan Yeung, JamesT. Kwok, Ana Fred, Fabio Roli, and Dick de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 163–172. Springer Berlin Heidelberg, 2006.

[25] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin Heidelberg, 1999.

[26] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, 27(7):950–959, June 2009.

[27] Tomas Toft. Sub-linear, secure comparison with two non-colluding parties. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 174–191. Springer Berlin Heidelberg, 2011.

[28] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

[29] Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, 2(1):25–36, August 2009.

A Description of Sub-Protocols

A.1 Encrypted Equality Test Protocol and Comparison Protocol

Given encryptions $\text{Enc}(y_1)$ and $\text{Enc}(y_2)$ of y_1 and y_2, respectively, where y_1 and y_2 are of ℓ-bit numbers in $\mathbb{Z}_N$. In our setting, the secure equality testing protocol outputs the encrypted value $\text{Enc}(b)$ where $b = 0$ if $y_1 \neq y_2$ and $b = 1$ if $y_1 = y_2$, without revealing y_1, y_2 and b where y_1 and y_2 are of ℓ bits. Our equality testing protocol is based on the idea of plaintext-space reduction introduced in [11], which can also be found in [18]. The setting of the equality check is different from the one proposed in [11]. In our case, the private key is shared between the parties, but in [11], one party holds the private key and the other party holds the encrypted numbers. We describe a secure encrypted equality test protocol based on plaintext-space reduction in Fig. 5. We use the greater-than protocol of Toft [27] with the modification that we replace the equality test protocol by π_{EQ}. We denote this protocol by π_{CMP} which takes inputs $\text{Enc}(x)$ and $\text{Enc}(y)$ and outputs $\text{Enc}(b)$ where $b = 1$ iff $x \geq y$ and $b = 0$, otherwise. Its round complexity is $O(\log(\ell))$ and computation complexity is $O(\ell \log(\ell) \log^*(\ell))$.

A.2 Shuffling and randomizing Protocol and Substitution Cost Protocol

Fig. 6 presents the protocol for computing the substitution cost for constructing the cost matrix.

Lemma 1. *The protocol π_{Sub} is secure in the presence of the semi-honest adversaries.*

Proof. The proof is based on the fact that π_{CMP} is secure and the messages received by the parties are semantically-secure threshold encryptions. □

B The Assignment Problem

Given a cost matrix $W = (w_{ij})_{n \times n}$, the assignment problem can be solved by the Hungarian algorithm with time complexity $O(n^3)$ [21]. The assignment problem can be viewed as a *perfect bipartite matching* in a complete bipartite graph. Before describing the main algorithm, we define some terms that will be used to describe the algorithm.

Let $G = (V, E)$ be a complete bipartite graph where $V = (X \cup Y)$ with $X \cap Y = \phi$ is the set of vertices and E is the set of edges, one vertex is connected to all other vertices. A *labeling* for the vertices of G, denoted by $\mathsf{lbl} : V \to \mathbb{R}$, is a function such that, for all $(u, v) \in E$, $\mathsf{lbl}_X(u) + \mathsf{lbl}_Y(v) \geq w(u, v)$ where $w(u, v) = w_{uv}$ is the weight of edge (u, v). The labeling of vertices of X and Y are denoted by $\mathtt{lbl}_X(\cdot)$ and $\mathtt{lbl}_Y(\cdot)$, respectively. An *equality subgraph* is defined as a subgraph $G_{\mathsf{lbl}} = (V, \mathcal{E}_{\mathsf{lbl}}) \subset (V, E)$ for which $\mathcal{E}_{\mathsf{lbl}} = \{(u, v) \in E : \mathsf{lbl}_X(u) + \mathsf{lbl}_Y(v) = w(u, v)\}$.

The neighbor of a vertex u, denoted by $N_{\mathsf{lbl}}(u)$, is defined as $N_{\mathsf{lbl}}(u) = \{v : (u, v) \in \mathcal{E}_{\mathsf{lbl}}\}$ and $N_{\mathsf{lbl}}(S) = \cup_{u \in S} N_{\mathsf{lbl}}(u)$. We denote the bipartite matching set by $\mathcal{M}$. We say a vertex v is *matched* if it is an endpoint of an edge in $\mathcal{M}$. Otherwise, v is free or unmatched. A path is

Protocol: `Equality Test` π_{EQ}

Input: Two encrypted numbers $\text{Enc}(y_1)$ and $\text{Enc}(y_2)$.

Output: $\text{Enc}(b)$ where $b = 0$ if $y_1 \neq y_2$ and $b = 1$ if $y_1 = y_2$.

1. Denote $y = y_1 - y_2$. P_1 and P_2 perform the operation: $\text{Enc}(y) = \frac{\text{Enc}(y_1)}{\text{Enc}(y_2)}$.
2. P_1 generates a random number A^1 and represents it in binary as $A^1 = A^1_{\ell-1}A^1_{\ell-2}...A^1_0$ and computes $c_1 \leftarrow \text{Enc}(y + A^1)$ and $C^1_i \leftarrow \text{Enc}(A^1_i), 0 \leq i \leq n-1$ and send $\{c_1, C^1_i, 0 \leq i \leq \ell - 1\}$ to P_2.
3. P_2 generates a random number A^2 and represents it in binary as $A^2 = A^2_{\ell-1}A^2_{\ell-2}...A^2_0$ and computes $c_2 \leftarrow \text{Enc}(y + A^1 + A^2)$ and computes $C^2_i \leftarrow \text{Enc}(A^2_i), 0 \leq i \leq n-1$ and send $\{c_2, C^2_i, 0 \leq i \leq \ell - 1\}$ to P_1.
4. P_1 and P_2 run π_{DistDec} to decrypt c_2 to obtain x where $x = y + A^1 + A^2 = y + A$ and $A = A^1 + A^2$.
5. P_1 and P_2 compute the ciphertexts $\text{Enc}(A_i)$ where $A = A_{\ell-1}A_{\ell-2}...A_0$ using $C^1_i, C^2_i, 0 \leq i \leq \ell - 1$ and additive circuits of two integers as follows. P_1 and P_2 computes $\text{Enc}(s_0)$ with $s_0 = 0$. For $i = 0$ to $\ell - 1$, P_1 and P_2 execute the following steps and for each encryption operation parties re-randomize the ciphertext:
   (a) P_1 computes $\text{Enc}(A^1_i s_i) = \text{Enc}(s_i)^{A^1_i}$ and $\text{Enc}(2A^1_i s_i) = \text{Enc}(s_i)^{2A^1_i}$ from $\text{Enc}(s_i)$ and send these two to P_2.
   (b) P_2 computes $\text{Enc}(A^2_i s_i) = \text{Enc}(s_i)^{A^2_i}$ and $\text{Enc}(2A^2_i s_i) = \text{Enc}(s_i)^{2A^2_i}$ from $\text{Enc}(s_i)$ and send these two to P_1.
   (c) Using $\text{Enc}(2A^2_i s_i)$, P_1 computes $\text{Enc}(2A^2_i A^1_i s_i) = \text{Enc}(2A^2_i s_i)^{A^1_i}$ and using $\text{Enc}(A^1_i s_i)$, P_2 computes $\text{Enc}(2A^1_i A^2_i s_i) = \text{Enc}(A^1_i s_i)^{A^2_i}$.
   (d) P_1 computes $\text{Enc}(s_{i+1}) = (C^2_i)^{A^1_i} \cdot \text{Enc}(A^1_i s_i) \cdot \text{Enc}(A^2_i s_i) = \text{Enc}(A^1_i A^2_i + A^1_i s_i + A^2_i s_i)$.
   (e) P_2 computes $\text{Enc}(s_{i+1}) = (C^1_i)^{A^2_i} \cdot \text{Enc}(A^1_i s_i) \cdot \text{Enc}(A^2_i s_i) = \text{Enc}(A^1_i A^2_i + A^1_i s_i + A^2_i s_i)$.
6. P_1 and P_2 independently perform the following steps:
   (a) Compute $\text{Enc}(A_i) = \frac{\text{Enc}(A^1_i) \cdot \text{Enc}(A^2_i) \cdot \text{Enc}(s_i) \cdot \text{Enc}_{pk}(4s_i A^1_i A^2_i)}{\text{Enc}(2A^1_i s_i) \cdot \text{Enc}(2A^2_i s_i)}$.
   (b) Compute $\text{Enc}(Z_i) = \frac{\text{Enc}(A_i) \cdot \text{Enc}(x_i)}{\text{Enc}(2x_i A_i)}$ using $\text{Enc}(A_i)$ and x_i where $Z_i = x_i \oplus A_i = x_i + A_i - 2x_i A_i$.
   (c) Compute $\text{Enc}(Z)$ as $\text{Enc}(Z) = \prod_{i=0}^{\ell-1} \text{Enc}(Z_i)$ where $Z = \sum_{i=0}^{\ell-1} Z_i$.
7. Set $\text{Enc}(y) \leftarrow \text{Enc}(Z)$, P_1 and P_2 execute Step 2 to Step 6 $\log^*(\ell)$ times.
8. P_1 and P_2 compute $\text{Enc}(1 - Z)$. Output $\text{Enc}(1 - Z)$.

Figure 5: Protocol for equality of two encrypted numbers using the Paillier threshold encryption scheme

Protocol: `Substitution Cost` π_{Sub}
Input: P_1's inputs Enc(C_1) and Enc(α_i); P_2's inputs Enc(C_2) and Enc(β_j);
Output: Enc(w_{ij}) where $w_{ij} = c(u_i \to v_j) = \min\{(c(u_i \to \epsilon) + c(\epsilon \to v_j)), |\alpha_i - \beta_j|\}$.

1. P_1 computes Enc(C_1) and Enc(α_i) and sends these to P_2.
2. P_2 computes Enc(C_2) and Enc(β_j) and sends these to P_1.
3. P_1 and P_2 compute Enc(C_1+C_2) = Enc($c(u_i \to \epsilon)+c(\epsilon \to v_j)$) by multiplying Enc($C_1$) and Enc($C_2$).
4. P_1 and P_2 run π_{CMP} with inputs Enc(α_i) and Enc(β_j) and obtain Enc(b).
5. P_1 first computes Enc($1-b$) from Enc(b) and then computes Enc($(1-b)\alpha_i$) and sends Enc($(1-b)\alpha_i$) to P_2.
6. P_2 first computes Enc($b-1$) from Enc(b) and then computes Enc($(b-1)\beta_j$) and sends Enc($(b-1)\beta_j$) to P_1.
7. Both parties compute Enc($|\alpha_i - \beta_j|$) = Enc($(1-b)\alpha_i$) $\cdot$ Enc($(b-1)\beta_j$).
8. P_1 and P_2 run π_{CMP} with inputs Enc($C_1 + C_2$) and Enc($|\alpha_i - \beta_j|$) and obtain Enc(b'). If $b' = 0$, output Enc(w_{ij}) = Enc($C_1 + C_2$)), otherwise output Enc($|\alpha_i - \beta_j|$).

Figure 6: Protocol for computing node substitution cost $c(u_i \to v_j)$

alternating if its edges alternate between $\mathcal{M}$ and $\mathcal{E}_{\mathsf{lbl}} - \mathcal{M}$. An alternating path is *augmenting* if both of its endpoints are free. Let P be an augmenting path with respect to $\mathcal{M}$ in $\mathcal{E}_{\mathsf{lbl}}$ and $\mathcal{P}$ be the set of edges in the path. The improved matching can be computed as $\mathcal{M}' = (\mathcal{M} \backslash \mathcal{P}) \cup (\mathcal{P} \backslash \mathcal{M})$. Algorithm 1 presents a pseudocode of the Hungarian algorithm.

C Graph Edit Operations and Cost Matrix

A standard set of graph edit operations are node *insertion* ($\epsilon \to u$), *deletion* ($u \to \epsilon$) and *substitution* ($u \to v$) and edge *insertion* ($\epsilon \to e$), *deletion* ($e \to \epsilon$) and *substitution* ($e_1 \to e_2$) and substitution of node and edge labels where ϵ denotes empty nodes or edges. The edge edit operations can be defined in terms of the node edit operations as follows. Let $e_1 = (u_1, u_2) \in E_1$ and $e_2 = (v_1, v_2) \in E_2$ where $u_1, u_2 \in V_1 \cup \{\epsilon\}$ and $v_1, v_2 \in V_2 \cup \{\epsilon\}$. An edge substitution operation between e_1 and e_2, denoted by $e_1 \to e_2$, is defined as the node substitution operations $u_1 \to v_1$ and $u_2 \to v_2$. If there is no edge e_1 in E_1 and $e_2 \in E_2$, then the edge insertion in G_1, denoted by ($\epsilon \to e_2$) is defined by $\epsilon \to v_1$ and $\epsilon \to v_2$. Similarly, if there is an edge $e_1 \in E_1$ and no edge e_2 in E_2, then the edge deletion, denoted by ($e_1 \to \epsilon$) is defined by $u_1 \to \epsilon$ and $v_2 \to \epsilon$.

The cost matrix is constructed by considering substitution costs of vertices and the costs of vertex insertions and deletions. The structure of the edit cost matrix $W = (w_{ij})_{(n+m)\times(n+m)}$

Algorithm 1 Hungarian algorithm based on perfect bipartite matching

```
1: Input: Graphs $G = (X, Y, E)$ and $W = (w_{i,j})$ where $X = \{u_1, u_2, ..., u_n\}$, $Y = \{v_1, v_2, ..., v_n\}$ and $w_{i,j} = wight(u_i, v_j)$
2: Output: Optimal assignment $\mathcal{M}$
3: procedure HUNGARIAN_BIPARTITE($G, W$)
4:     Find a valid labeling $\texttt{lbl}$: Set $\texttt{lbl}(v) = 0, v \in Y$ and $\texttt{lbl}(u) = \max_{v' \in Y}\{w(u, v')\}, u \in X$.
5:     $\mathcal{M} \leftarrow \phi$ //Finding initial matching $\mathcal{M}$ corresponding to the labeling $\texttt{lbl}$
6:     while No more edges can be added do
7:         Select an edge $(u, v)$ where none of $u$ and $v$ is incident to an edge in $\mathcal{M}$
8:         $\mathcal{M} \leftarrow \mathcal{M} \cup (u, v)$.
9:     end while
10:    if $\mathcal{M}$ is perfect, i.e., $|\mathcal{M}| = n$ then
11:        return $\mathcal{M}$
12:    else
13:        Pick free vertex $u \in X$ and set $S \leftarrow \{u\}$ and $T \leftarrow \Phi$
14:        if $N_{\texttt{lbl}}(S) = T$ then
15:            $\delta_{\texttt{lbl}} = \min_{u \in S, v \notin T}\{\texttt{lbl}(u) + \texttt{lbl}(v) - w(u, v)\}$
16:            Update labeling $\texttt{lbl}' \leftarrow \texttt{lbl}$ where
               $$\texttt{lbl}'(s) = \begin{cases} \texttt{lbl}(s) - \delta_{\texttt{lbl}} & \text{if } s \in S \\ \texttt{lbl}(s) + \delta_{\texttt{lbl}} & \text{if } s \in T \\ \texttt{lbl}(s) & \text{otherwise.} \end{cases}$$
17:        else
18:            Pick $v \in N_{\texttt{lbl}}(S) - T$
19:            If $v$ is unmatched, find an augmenting path $P := u - v$. Augment $\mathcal{M}$ as $\mathcal{M}' = \mathcal{M}\Delta\mathcal{P}$ and set $\mathcal{M} = \mathcal{M}'$ and Go to 10.
20:            If $v$ is a matched, say to $z$, extend alternating tree: $S = S \cup \{z\}, T = T \cup \{v\}$. Go to Step 14.
21:        end if
22:    end if
23:    return $\mathcal{M}$
24: end procedure
```

has the following form [26]:

$$C = \left[\begin{array}{cccc|cccc}
w_{11} & w_{12} & \cdots & w_{1m} & w_{1\epsilon} & \infty & \cdots & \infty \\
w_{21} & w_{22} & \cdots & w_{2m} & \infty & w_{2\epsilon} & \cdots & \infty \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
w_{n1} & w_{n2} & \cdots & w_{nm} & \infty & \infty & \cdots & w_{n\epsilon} \\
\hline
w_{\epsilon 1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\
\infty & w_{\epsilon 2} & \cdots & \infty & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\infty & \infty & \cdots & w_{\epsilon m} & 0 & 0 & \cdots & 0
\end{array}\right] = \left[\begin{array}{cc} W_1 & W_2 \\ W_3 & W_4 \end{array}\right]$$

where the submatrix W_1 is corresponding to the cost assignment of nodes $(i \rightarrow j)$, W_2 and W_3 are corresponding to the cost assignment of node deletion $(i \rightarrow \epsilon)$ and insertion $(\epsilon \rightarrow i)$ of nodes. The insertion and deletion of edges are not taken care of in the cost matrix. However, it is not hard to incorporate the edge substitution cost into the matrix entries. We don't provide the details here.