Link Analysis for Private Weighted Graphs

Jun Sakuma
University of Tsukuba
1-1-1 Tennodai,
Tsukuba, Japan
jun@cs.tsukuba.ac.jp

Shigenobu Kobayashi
Tokyo Institute of Technology
4259 Nagatsuta-cho
Yokohama, Japan
kobayasi@dis.titech.ac.jp

ABSTRACT

Link analysis methods have been used successfully for knowledge discovery from the link structure of mutually linking entities. Existing link analysis methods have been inherently designed based on the fact that the entire link structure of the target graph is observable such as public web documents; however, link information in graphs in the real world, such as human relationship or economic activities, is rarely open to public. If link analysis can be performed using graphs with private links in a privacy-preserving way, it enables us to rank entities connected with private ties, such as people, organizations, or business transactions. In this paper, we present a secure link analysis for graphs with private links by means of cryptographic protocols. Our solutions are designed as privacy-preserving expansions of well-known link analysis methods, PageRank and HITS. The outcomes of our protocols are completely equivalent to those of PageRank and HITS. Furthermore, our protocols theoretically guarantee that the private link information possessed by each node is not revealed to other nodes.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

link analysis, privacy, ranking, HITS, PageRank

1. INTRODUCTION

Link-based analysis has been developed in the form of algorithms that discover useful information from the link structure of mutually linking entities. In particular, HITS [7] and PageRank [9] have been successfully used for the ranking of hyperlinked web documents. These link analysis methods were originally designed for the analysis of web documents; however, these can be readily applied to mutually linking entities, such as referenced academic papers, protein-protein interactions, and so on.

SIGIR'09, July 19–23, 2009, Boston, Massachusetts, USA.

In general, link analysis methods take the entire link structure as its input. Indeed, for the computation of Google's PageRank, the linking structures of web documents are collected by crawling agents which actually wander around public web documents. The same holds for citation graphs of academic papers or interaction graphs of protein networks. As shown, existing link analysis methods have inherently been designed based on the fact that the entire link structure of the target graph is observable; however, link information in the real world, such as human relationships or economic activities, is rarely open to public.

In this paper, we present link analysis solutions for graphs of privately connected entities. Let there be a directed weighted graph $G = (V, E, W)$ where V is a set of vertices, E is a set of edges, and W is a weight matrix. Throughout this paper, we assume that the set of vertices corresponds to a collection of distributed nodes where the computational power of each node is polynomial. Edges correspond to links between nodes; weights of edges correspond to weights of these links. Let there be a link of node i pointing to node j. In our setting, we assume that link e_{ij} and weight of the link w_{ij} are not desired to be known by nodes other than node i and node j. Furthermore, we design our link analysis solutions based on the three privacy models of graphs described as below:

Weight-aware model. If both the head node i and the tail node j know the existence of the link and the weight value, this is designated as *weight-aware link-aware* model (or weight-aware model for short). For example, consider commercial relationships among enterprises. Each enterprise may conduct business transactions with the other enterprises. Let the ith enterprise purchase some products from the jth enterprise. This transaction corresponds to link e_{ij} and the transaction value corresponds to weight w_{ij}. In this case, both the ith and jth enterprise are aware of the existence of this link and know the weight value, but enterprises other than i and j do not know the existence of this transaction and the transaction value.

Link-aware model. If the head node i and the tail node j know the existence of the link, but the weight value is only known by the head node i, this is designated as *link-aware weight-unaware* model (or link-aware model for short). For example, consider call logs of cell-phones. Let caller i make a phone call to receiver j. This call corresponds to link e_{ij} and the probability that i makes a phone call to j corresponds to the weight w_{ij} of e_{ij}. In this case, both caller i and receiver j are aware of the existence of the link, but the call probability w_{ij} are known only by caller i.

Link-unaware model. If only the head node i knows the existence of the link and the weight value, but the tail node j knows nothing, this is designated as *link-unaware weight-unaware* model (or link-unaware model for short). For example, consider a peer evaluation scheme among members of personnel. Each member can choose a limited number of other members and provide eval-

Table 1: Comparison of computational complexity and privacy preservation. n: the number of nodes, Δ: the maximum degree of nodes, ℓ, the number of hops for message routing. The measure of the privacy is discussed in Section 6.

	Link-aware		Link-unaware	
	complexity	privacy	complexity	privacy
DSA [6]	$O(\Delta)$	bad	—	—
PPLA [3]	—	—	$O(n^2)$	medium
SFE [11, 8]	polynomial	good	polynomial	good
proposal	$O(\Delta)$	good	$O(\Delta + \ell)$	good

uation values. Let the ith member evaluate the jth member. This evaluation relationship corresponds to link e_{ij} and the evaluation value corresponds to weight w_{ij}. In this case, only the ith member is aware of the existence of the link and the evaluation value, but the jth member knows nothing.

We designate graphs with private links and private weights as *private graphs* or *private weighted graphs*. Privacy models shown above are typical classes of private weighted graphs. Usually, links or social ties, such as who buys from whom, who calls whom, or who evaluates whom, are not preferred to be revealed in public due to confidentiality or privacy concerns. If link analysis can be performed using private weighted graphs without violating the privacy model of the graph, valuable information would be extracted from privately networked entities. This contributes to enhance information retrieval technologies for networked entities in the real world. Considering this, the main objective of this paper is to present secure protocols for link analysis with private weighted graphs.

Related works. If a trusted third party (TTP) which never deviates from the specified protocol and does not reveal any auxiliary information is available, link analysis can be readily and securely achieved; however, facilitating such a party is unrealistic in general. Therefore, we aim to design a protocol without using any TTP. In this subsection, we compare three existing solutions designed without a TTP. Table 1 provides a qualitative comparison of link analysis solutions in terms of computational complexity and privacy preservation.

Kempe et. al. have presented a decentralized spectral analysis (DSA) for computing the top k eigenvectors of a weighted adjacency matrix distributed over peer-to-peer networks [6], which can be readily used for PageRank. The computation of their protocol is well decentralized and scales well even with large-scale networks; if DSA is used for the computation of PageRank, the computational complexity is $O(\Delta)$ at each step at each node where Δ is the maximum degree of the graph ($\Delta \ll n$ in sparse graphs). Their solution is not designed for the purpose of privacy preservation originally, but the privacy of links and weights are preserved to some extent because private information is not directly exchanged between nodes and the communication occurs only between nodes connected with links. Nonetheless, there is no guarantee of privacy preservation because private information might be inferred from messages exchanged during the protocol execution.

Duan et. al. have presented a privacy-preserving link analysis (PPLA) using HITS [3] in the link-unaware model. The idea of their solution is based on the homomorphism of a public-key cryptosystem, which allows us to compute encrypted summation of integers without knowing summands. The limitation of their protocol is two-fold: (1) The protocol uses a special node called tallier, which collects encrypted values from all nodes and compute the encrypted summation of them by means of the homomorphic property. Due to this design, their solution is not fully decentralized and includes a step whose computational complexity is $O(n^2)$. Since we assume that hundreds or thousands of nodes participate in the protocol, this causes inefficiency. (2) Their protocol allows the nodes to reveal intermediate values of authority in the middle, which might enable neighboring nodes to infer private links or weights. Thus, their privacy preservation is not perfect as well.

Secure function evaluation (SFE) [11, 4] is a general and well studied methodology for evaluating any distributed computation privately; this can be a solution for link analysis in both link-aware and link-unaware model. Although the computational complexity of SFE is polynomially bounded and the privacy preservation with this solution can be perfect, these computations can be too inefficient for practical use, particular when the input size is large. We discuss these topics with experimental results in Section 6 again.

Our contribution. As discussed, link analysis solutions with both good scalability and perfect privacy-preservation have not been presented previously. In this paper, we formally state the problems of secure link analysis in both the link-aware model and the link-unaware model (Section 3). Our main contribution is as follows.

We propose secure spectral ranking (SSR) as a secure link analysis method for link-aware/link-unaware private weighted graphs (Section 5). As variations of SSR, we present PrivateRank and PrivateHITS, too. These are secure expansions of well-known link analysis methods, PageRank and HITS.

Our solution theoretically guarantees privacy preservation of private weighted graphs. Especially in the solution for the link-aware model, it is guaranteed that nothing but the final output of the spectral ranking is revealed. In the link-unaware model, the solution allows each node to know the number of links pointing to that node as auxiliary information. Nevertheless, our solutions do not disclose any intermediate messages and achieve more rigorous security compared to existing solutions. Furthermore, the computational complexity of our secure spectral ranking is $O(\Delta)$ in the link-aware model as well as DSA. In the link-unaware model, the computational complexity is $O(\Delta+\ell)$ where ℓ is the number of hops required for anonymous message transfer (Section 4.2). In our solution, the computation is decentralized and distributed equally to computational nodes, so is scalable compared to existing solutions.

Comparisons between our solutions and existing solutions are also summarized in Table 1. In order to demonstrate the efficiency of our protocol, the results of computational experiments with the complexity analysis are shown in Section 6.

2. LINK ANALYSIS

In these section, we introduce a basic link analysis method, spectral ranking. Then two well-known link analysis ranking methods, PageRank [9] and HITS [7] are introduced.

Let $G = (V, E, W)$ be a weighted directed graph, where $V = \{1, ..., n\}$ is a set of vertices and $E = \{e_{ij}\}$ is a set of edges. The edge between two vertices i and j has a non-negative weight $w_{ij} \geq 0$. $W = (w_{ij})$ is designated as a *weight matrix*. $w_{ij} = 0$ means that there is no edge between i and j. The *degree* of vertex i is defined as $d_i = \sum_{j \in V} w_{ij}$. The maximum degree is defined as $\Delta = \max_i d_i$. Let $D = \mathrm{diag}(d_1, ..., d_n)$ be the *degree matrix*. The adjacency matrix $A = (a_{ij})$ is defined as

$$a_{ij} = \begin{cases} 1 \text{ if } e_{ij} \in E \\ 0 \text{ o.w.} \end{cases} . \quad (1)$$

Let $N_{out}(i) = \{j | j \in V, e_{ij} \in E\}$ and $N_{in}(i) = \{j | j \in V, e_{ji} \in E\}$. $N_{out}(i)$ (resp. $N_{in}(i)$) is a set of vertices which has an edge pointing from (resp. pointing to) vertex i.

Given a graph, link analysis provides some scores of each node by exploiting the link structure.

2.1 Spectral Ranking and PageRank

Consider a Markov chain in which a jump from node i to node j occurs with probability p_{ij}. Given a weighted graph, the probability transition matrix $P = (p_{ij})$ is defined as $P = D^{-1}W$. The stationary distribution of a Markov chain is defined as the visiting probability of each state $\lim_{t\to\infty} \frac{\eta(i,t)}{t}$ where $\eta(i,t)$ is the number of visits to state i in t steps. From the nature of the transition probability matrix, the stationary distribution $\boldsymbol{x} = (x_1, ..., x_n)^T$ satisfies

$$\boldsymbol{x}^T = \boldsymbol{x}^T P \tag{2}$$

where $\sum_i x_i = 1$ and $x_i \geq 0$ for all i. If the Markov chain is ergodic, the largest eigenvalue is always 1 and the principal eigenvector (the eigenvector paired with the largest eigenvalue) of P becomes the stationary distribution of the Markov chain. *Spectral ranking* evaluates the importance of nodes by the visiting probability.

PageRank makes use of the stationary distribution to evaluate the importance of hyperlinked web documents. The idea behind this is that documents referred from documents with a higher PageRank are then themselves given a higher PageRank. Given a graph of web documents, the transition probability matrix

$$P = (1-\epsilon)D^{-1}A + \frac{\epsilon}{n}\mathbf{1}\mathbf{1}^T \tag{3}$$

is considered, where $\mathbf{1}$ is the vector whose elements are all 1. This transition matrix assumes a random walk by a user who randomly chooses a hyperlink in the current document and jumps to the next document with probability $1-\epsilon$. Otherwise, the user jumps to a document chosen randomly from the whole document set with probability ϵ (teleportation probability). Teleportation probability helps to guarantee the ergodicity of the Markov chain.

As represented in eq. 2, solving stationary distributions is reduced to eigenproblems of P. In this case, we need only the left principal eigenvector. Power iteration is commonly used for this purpose. The initial vector is initialized s.t. $\sum_i x_i^{(0)} = 1$ and then, the following updates are iterated until convergence

$$(\boldsymbol{x}^{(t)})^T \leftarrow P(\bar{\boldsymbol{x}}^{(t-1)})^T, \quad \bar{\boldsymbol{x}}^{(t)} \leftarrow \frac{\boldsymbol{x}^{(t)}}{\|\boldsymbol{x}^{(t)}\|}, \tag{4}$$

where $\|\cdot\|$ denotes 2-norm. Usually, power iteration requires normalization of $\boldsymbol{x}^{(t)}$; however, when P is a probability matrix and $\boldsymbol{x}$ is a probability vector, the normalization step can be skipped. The convergence property is shown as follows. The proof is omitted.

Lemma 1. Let $\boldsymbol{x}$ be the principal eigenvector of P. Let $\boldsymbol{x}^{(t)}$ be the eigenvector obtained by power iteration after t updates. Then,

$$\|\boldsymbol{x}^{(t)} - \boldsymbol{x}\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^t\right), \tag{5}$$

where λ_1 and λ_2 denotes the largest and the second largest eivenvalues, respectively.

2.2 HITS

Kleinberg has proposed a different link analysis HITS which evaluates the importance of mutually linked documents with the value of hub and the value of authority. The value of hub is higher if the node has more links to authoritative documents. The value of authority is higher if it is linked from nodes with higher value of hub. Let $\boldsymbol{a}$ and $\boldsymbol{h}$ be n dimensional vectors of the value of authority and the value of hub, respectively. Then, with the adjacency matrix A, HITS iterates the following update equations alternately:

$$\boldsymbol{a}^{(t)} \leftarrow A^T\boldsymbol{h}^{(t-1)}, \quad \boldsymbol{h}^{(t)} \leftarrow A\boldsymbol{a}^{(t)} \tag{6}$$

Since $A^T\boldsymbol{h}^{(t-1)} = A^TA\boldsymbol{a}^{(t-1)}$ and $A\boldsymbol{a}^{(t)} = AA^T\boldsymbol{h}^t$, HITS is reduced to eigenproblems of A^TA and AA^T, as spectral ranking is.

3. PROBLEM STATEMENT

Given a graph with n nodes, we assume that the elements of the adjacency matrix and the weight matrix are partitioned into (not necessarily disjoint) n parts. Each node possesses a mutually different part of the matrix privately. We begin with the definitions of the matrix partitioning model. Then, three types of private weighted graphs are defined as typical privacy models of graphs. Based on these definitions, we state problems of secure link analysis.

3.1 Private Weighted Graphs

Let $M = (m_{ij})$ be a $n \times n$ matrix. We write the ith row of M as $\boldsymbol{m}_{i*}$ and the ith column of M as $\boldsymbol{m}_{*i}$. Let there be n nodes. M are partitioned into n parts and each node privately holds one of the parts. In this paper, we specifically focus on the following two typical partitioning patterns.

Definition 1. (Row private) Let there be a $n \times n$ matrix M and n parties. For all i, if the ith party knows a row vector $\boldsymbol{m}_{i*}$, but does not know other row vectors $\boldsymbol{m}_{p*}(p \neq i)$ of M, then M is *row private*.

Definition 2. (Symmetrically private) Let there be a $n \times n$ matrix M and n parties. For all i, if the ith party knows $\boldsymbol{m}_{i*}$ and $\boldsymbol{m}_{*i}$, but does not know other elements m_{pq} where $p, q \neq i$, then M is *symmetrically private*.

Based on these matrix partitioning models, three types of privacy models of graph, *private weighted graphs*, are stated.

Definition 3. (Weight-aware private weighted graph) Let $G = (V, E, W)$ where V is a collection of computational nodes. If the adjacency matrix A (associated with E) and the weight matrix W are symmetrically private, G is a weight-aware private weighted graph.

Definition 4. (Link-aware private weighted graph) Let $G = (V, E, W)$ where V is a collection of computational nodes. If the adjacency matrix A (associated with E) is symmetrically private and the weight matrix W is row private, G is a link-aware private weighted graph.

When adjacency matrix A is symmetrically private, node i knows $N_{in}(i)$ and $N_{out}(i)$, i.e., node i knows who links to i and who is linked to from i. The same holds for the weight matrix. In the link-aware model, although node i knows who links to node i, node i cannot know the weight value w_{ji}.

Definition 5. (Link-unaware private weighted graph) Let $G = (V, E, W)$ where V is a collection of computational nodes. If the adjacency matrix A (associated with E) and the weight matrix W are row private, G is a link-unaware private weighted graph.

In this paper, we treat weight-aware private weighted graphs as link-aware private weighted graphs since the former is the special case of the latter. The reason is described in Section 3.3.

3.2 Secure Link Analysis

Let $f : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times d}$ be a function which computes scores for link analysis, in which f takes the weight matrix $W \in \mathbb{R}^{n\times n}$ as the input and returns the score matrix $X \in \mathbb{R}^{n\times d}$ as the output. Given a private weighted graph, the problems of secure link analysis are stated as follows.

Statement 1. Secure link analysis for link-aware (resp. link-unaware) private weighted graph: Let there be link-aware (resp. link-unaware) private weighted graph $G = (V, E, W)$. After the execution of secure link analysis among nodes, $f(W) = X$ is correctly evaluated and X is distributed such that X is row private among nodes. Furthermore, each node learns nothing else and G is kept link-aware (resp. link-unaware).

Intuitively, if the link analysis is secure in the sense of Statement 1, what the ith node can learn is limited to the ith row of score matrix X and nothing else. In the link-aware/link-unaware model, W and A need to be kept symmetrically/row private after the execution of the protocol. In Section 5, we specifically present protocols of spectral ranking that are secure and correct in the sense of Statement 1. Then, expansions to PageRank and HITS are shown.

3.3 Our Approach

Most of our discussions in this subsection are true of spectral ranking, PageRank, and HITS, so we restrict our attention to spectral ranking for now.

Link-aware model: Let the ith node hold x_i and update it in power iteration. The update of the ith element is written as $x_i^{(t)} \leftarrow \sum_{i=1}^{n} x_j^{(t-1)} p_{ji}$. In our case, if $j \notin N_{in}(i)$, $p_{ji} = 0$. So,

$$x_i \leftarrow \sum_{j \in N_{in}(i)} x_j p_{ji}, \tag{7}$$

holds. This means that ith node can update $x_i^{(t)}$ if ith node can receive $x_j^{(t-1)} p_{ji}$ from node j for all $j \in N(i)_{in}$. The problem here is that both x_j and p_{ji} need to be kept private in the link-aware model. To update this without node i being able to know x_j and p_{ji}, we introduce a public-key cryptography with homomorphic property, which allows any node to add two ciphers without decryption. Node j encrypts x_j and p_{ji} in a way that node i cannot decrypt, and send them to node i. With these, node i can update eq. 7 without learning $j(\in N_{in}(i))$'s information by means of the homomorphism. The weight-aware model can be equally treated.

Link-unaware model : The significant difference between the weight/link-aware model and the link-unaware model is in the nature of the communication. Let there be a link of node j pointing to node i. In the link-aware model, node j is allowed to establish a direct connection with node i because the connection does not violates the given privacy model. On the other hand, if the same protocol is executed in the link-unaware model, a request for a connection to node i itself reveals to node i that a link of node j is pointing to node i. Thus, in the link-unaware model, the problem is not only in the secrecy of messages but also in the anonymity of connections. For this, we make use of the concept of *onion routing*, which provides anonymous connection over a public network.

In next section, we introduce two tools, homomorphic cryptography and onion routing.

4. BUILDING BLOCKS

4.1 Homomorphic Public-key Cryptosystem

In a public key cryptosystem, encryption uses a *public key* that can be known to everyone, while decryption requires knowledge of the corresponding *private key*. Given a corresponding pair of $(\mathsf{sk}, \mathsf{pk})$ of private and public keys and a message m, then $c = \mathrm{Enc}_{\mathsf{pk}}(m; \ell)$ denotes a (random) encryption of m, and $m = \mathrm{Dec}_{\mathsf{sk}}(c)$ denotes decryption. The encrypted value c uniformly distributes over $\mathbb{Z}_N$ if ℓ is taken from $\mathbb{Z}_N$ randomly. An *additive homomorphic cryptosystem* allows addition computations on encrypted values without knowledge of the secret key. Specifically, there is some operation $\cdot$ (not requiring knowledge of sk) such that for any plaintexts m_1 and m_2,

$$\mathrm{Enc}_{\mathsf{pk}}(m_1 + m_2; \ell) = \mathrm{Enc}_{\mathsf{pk}}(m_1; \ell_1) \cdot \mathrm{Enc}_{\mathsf{pk}}(m_2; \ell_2), \tag{8}$$

where ℓ is uniformly random provided that at least one of ℓ_1 and ℓ_2 is. Based on this property, it also follows that given a constant k and the encryption $\mathrm{Enc}_{\mathsf{pk}}(m_1; \ell)$, we can compute multiplications by k via repeated application of $\cdot$. This also enables a re-randomization property, which allows the computation of a new random encryption $c' = \mathrm{Enc}_{\mathsf{pk}}(m; \ell')$ of m from an existing encryption $c = \mathrm{Enc}_{\mathsf{pk}}(m; \ell)$ of m, again without knowledge of the private key or of m, as follows:

$$\mathrm{Enc}_{\mathsf{pk}}(m; \ell) = \mathrm{Enc}_{\mathsf{pk}}(m; \ell_1) \cdot \mathrm{Enc}_{\mathsf{pk}}(0; \ell_2). \tag{9}$$

In the rest of the paper, we omit the random number ℓ from our encryptions for simplicity.

Threshold cryptosystem. In an (m, t)-*threshold cryptosystem*, m nodes share a common public key pk while the nodes hold different private keys $\mathsf{sk}^1, ..., \mathsf{sk}^n$. Each node can encrypt any message with the common public key. Decryption cannot be performed by fewer than t nodes, and can be performed by any group of at least t nodes using a recovery algorithm based on the public key and their *decryption shares* $\mathrm{Dec}_{\mathsf{sk}^1}(c), ..., \mathrm{Dec}_{\mathsf{sk}^n}(c)$. We require a cryptosystem that provides semantic security (under appropriate computational hardness assumptions), re-randomization, the additive homomorphic property, and threshold decryption, such as the generalized Paillier cryptosystem [1].

4.2 Onion Routing

Assume that sender node $s \in V$ wants to send a message $m \in \mathbb{Z}_N$ to receiver node $r \in V$, but the sender does not want the receiver to identify who is the sender (sender anonymity). The onion routing network allows the connection between the sender and receiver to remain anonymous [5]. In the onion routing network, each node (called *onion router*) has a different key pair of a (not necessarily homomorphic and threshold) public-key cryptosystem.

The message transferred by onion routing is prepared as follows. Let $o_1, ..., o_\ell$ be onion routers chosen by the sender. The message is sent from the sender s, travels through $o_1, ..., o_\ell$, and arrives at the receiver r. Let pk^r be the receiver's public key and pk^h be the public key of the hth onion router on the route $(h = 1, ..., \ell)$. The sender prepares and sends a message (called an *onion*) as follows:

1. Computes $c_r = \mathrm{Enc}_{\mathsf{pk}_r}(0)$
2. Computes $c_\ell = (\mathrm{Enc}_{\mathsf{pk}^\ell}(r), \mathrm{Enc}_{\mathsf{pk}^\ell}(c_r))$
3. For $h = \ell - 1, ..., 1$, computes $c_h = (\mathrm{Enc}_{\mathsf{pk}^h}(o_{h+1}), \mathrm{Enc}_{\mathsf{pk}^h}(c_{h+1}))$
4. Sends onion $(c_1, \mathrm{Enc}_{\mathsf{pk}^r}(m))$ to o_1

The onion has a layered data structure where the hth layer includes the address of the $(h+1)$th onion router encrypted with the hth onion router's public key. Thus, o_h can learn o_{h+1} and o_{h-1} (because the onion is sent from o_{h-1}), but cannot learn the addresses of the other onion routers on the route because these are encrypted in such a way that o_h cannot decrypt them. Furthermore, o_h cannot learn whether o_{h+1} is the receiver or not. Thus, the sender anonymity is preserved.

The onion is transferred to the receiver as follows. o_h receives $(c_h, \mathrm{Enc}_{\mathsf{pk}^r}(m))$ and learns o_{h+1} from $\mathrm{Dec}_{\mathsf{sk}^h}(c_h)$. If o_{h+1} is 0, then o_h learns that she is the receiver and obtains the message as $m = \mathrm{Dec}_{\mathsf{sk}^r}(\mathrm{Enc}_{\mathsf{pk}^r}(m))$. Otherwise, o_h learns o_{h+1} and $(c_{h+1}, \mathrm{Enc}_{\mathsf{pk}^r}(m))$ is sent to o_{h+1}. The message m is learned only by the receiver because it is encrypted by the receiver's public key.

5. SECURE LINK ANALYSIS

In this section, as secure link analysis, secure spectral ranking (SSR) in the link-aware model is presented. Then, the protocol is expanded to PageRank, HITS, and protocols for the link-unaware case. The detail of SSR is described in Figure 1.

5.1 Secure Spectral Ranking in Link-aware Model

5.1.1 Setup and Initialization

For secure spectral ranking, the computation of eq. 7 is performed by means of a homomorphic public-key cryptosystem. We assume that all nodes agree to make use of a common key set $\mathcal{K} = \{\mathsf{pk}, \mathsf{sk}^i, ..., \mathsf{sk}^n\}$ of the threshold homomorphic public-key cryptosystem. sk^i is held only by the ith node; pk is commonly known to all nodes. The key set is securely prepared by means of distributed key generation schemes. See [2] for example.

In the first step, probability matrix $P = D^{-1}W$ is prepared. $p_{ij} = w_{ij}/d_i$ is locally computed from row private W. Since typical public-key cryptosystems take only integers as inputs, p_{ij} is magnified such that $b_{ij}(= Lp_{ij}) \in \mathbb{Z}_N$ with some sufficiently large L. In regular power iteration, the stationary distribution is initialized to be a probability vector. Instead, in the second step of this protocol, $\boldsymbol{q}$ is initialized such that $\sum_i q_i = K$ with sufficiently large K and $q_i \in \mathbb{Z}_N$ for all i for the same reason. Each node can simply initialize as $q_i^{(0)} \leftarrow 1$. In this case, $K = n$.

5.1.2 Power Iteration

In the third step, power iteration is performed, in which node i is responsible for the update of the ith element of $\boldsymbol{q}$.

The following lemma shows the remaining steps correctly compute the principal eigenvector of P.

Lemma 2. Let $\boldsymbol{\pi}^*$ be the stationary distribution of P. The output of secure spectral ranking $\boldsymbol{\pi}^{(t)}$ is a probability vector such that

$$\|\boldsymbol{\pi}^* - \boldsymbol{\pi}^{(t)}\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^t\right), \tag{10}$$

where λ_1 and λ_2 are the largest and the second-largest eigenvalues of P, respectively.

Proof. Consider updates in step 3(b). By the homomorphic property of the cryptosystem, the update is rearranged as:

$$\mathrm{Enc}_{\mathsf{pk}}(q_i^{(t)}) \leftarrow \prod_{j \in N_{in}(i)} \left(\mathrm{Enc}_{\mathsf{pk}}(q_j^{(t-1)})^{b_{ji}}\right) = \mathrm{Enc}_{\mathsf{pk}}\left(\sum_{j \in N_{in}(i)} b_{ji} q_j^{(t-1)}\right). \tag{11}$$

Note that if both sides of eq. 11 could be decrypted,

$$q_i^{(t)} \leftarrow \sum_{j \in N_{in}(i)} \left(b_{ji} q_j^{(t-1)}\right) \tag{12}$$

is obtained, which configures power iteration with B. Thus, the updates with eq. 11 and with eq. 7 are equivalent except that values are magnified, encrypted, and not normalized in eq. 11.

Let $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, ...)$ be the sequence of regular power iteration with P. If $\boldsymbol{x}^{(0)}$ is initialized as a probability vector, $\boldsymbol{x}^{(t)}$ is a probability vector, too. In the sequence $(\boldsymbol{q}^{(0)}, \boldsymbol{q}^{(1)}, ...)$ of eq. 12, $\boldsymbol{q}^t$ and $\boldsymbol{x}^t$ are parallel since $B = LP$, but the size is different since $\boldsymbol{q}^{(t)}$ is not normalized. The summation $\sum_{i=1}^n q_i^{(t)}$ is

$$\sum_{i=1}^n q_i^{(t)} = \sum_{i=1}^n \sum_{j \in N_{in}(i)} b_{ji} q_i^{(t-1)} = L \sum_{i=1}^n q_i^{(t-1)}. \tag{13}$$

From $\sum_{i=1}^n q_i^{(0)} = K$ and eq. 13, we obtain

$$\sum_{i=1}^n q_i^{(t)} = KL^{t-1} \tag{14}$$

by the inductive method. Therefore, $\boldsymbol{x}^{(t)} = \boldsymbol{q}^{(t)}/KL^{t-1}$ holds. Consequently, from Lemma 1, it is concluded that $\boldsymbol{q}^{(t)}/KL^{t-1}$ converges to the stationary distribution of P. Furthermore, the convergence is geometric

$$\left\|\boldsymbol{\pi}^* - \frac{\boldsymbol{q}^{(t)}}{KL^{t-1}}\right\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^t\right). \tag{15}$$

Thus, the output of the protocol, $\boldsymbol{\pi}^{(t)} = \frac{1}{KL^{t-1}}(q_1^{(t)}, ..., q_n^{(t)})^T$ converges to the stationary distribution of P. □

Procedure LinkAwareSecureSpectralRanking (W, K, L)

- Public input: $K \in \mathbb{Z}_N$, $L \in \mathbb{Z}_N$ s.t. $Lp_{ij} \in \mathbb{Z}_N$ for all i, j
- Private input of node i: $\boldsymbol{w}_{i*}$ (row private W)
- Key setup: All nodes jointly generate a key set $\mathcal{K} = \{\mathsf{pk}, \mathsf{sk}^i, ..., \mathsf{sk}^n\}$ in a distributed way so that pk is commonly known to all nodes and sk^i is possessed only by the ith node.

1. (Setup of P) Agent i computes:
   (a) $d_i \leftarrow \sum_{j \in N_{out}(i)} w_{ij}$
   (b) for all j, $p_{ij} \leftarrow w_{ij}/d_i$, $b_{ij} \leftarrow Lp_{ij}$
2. (Initialization) Agent i updates
   $q_i^{(0)} \leftarrow K_i$ s.t. $\sum_i K_i = K$ and $t \leftarrow 1$.
3. (Power iteration) Node i computes the following steps until convergence:
   (a) For all $k \in N_{out}(i)$, node i computes
   $c_{ik,\mathsf{pk}} \leftarrow (\mathrm{Enc}_{\mathsf{pk}}(q_i^{(t-1)}))^{b_{ik}}$ and sends $c_{ik,\mathsf{pk}}$ to node k
   (b) For all $j \in N_{in}(i)$, node i receives $c_{ji,\mathsf{pk}}$ and computes $\mathrm{Enc}_{\mathsf{pk}}(q_i^{(t)}) \leftarrow \prod_{j \in N_{in}(i)} c_{ji,\mathsf{pk}}$, $t \leftarrow t + 1$
   (c) Node i and $j \in_r N(i)$ performs convergence detection protocol. If convergence is not detected, broadcast "no convergence". If no broadcast message received, jump to step 4. Else, jump to step 3(a).
4. (Decryption) Node i and $N_{out}(i)$ run the recover scheme and obtain $q_i^{(t)}$. Then, output $\pi_i^{(t)} = q_i^{(t)}/KL^{t-1}$

Figure 1: Secure Spectral Ranking in the Link-aware Model

5.1.3 Normalization and Convergence Detection

As mentioned, secure function evaluation (SFE) is a cryptographic primitive which allows two or more parties to evaluate a specified function of their inputs without revealing (anything else about) their inputs to each other [4, 11]. We make use of SFE for normalization and convergence detection.

From Lemma 2, the normalization of $\boldsymbol{q}^{(t)}$ seems to be not required in the middle of the protocol; however, if $q_i^{(t)}$ is greater than N, the addition of encrypted integers does not work correctly because the cryptography is modulo N arithmetic. Since $\sum_i q_i^{(t)}$ increases geometrically, normalization is required to avoid overflow. In order to normalize encrypted $\boldsymbol{q}^{(t)}$ securely, we make use of SFE for *private division* [10], which allows us to divide encrypted integers by a given specific integer (in our case KL^{t-1}). Since $N = 2^{1024}$ in typical cryptography settings, the normalization does not have to be invoked so often. For convergence detection, whether $|q_i^{(t)} - q^{(t-1)}| > \theta$ is true needs to be checked securely at each node with a given threshold parameter θ. This is readily realized by SFE for *private comparison* of $q_i^{(t)}$ and $q^{(t-1)}$ at each node.

5.2 Security

We assume nodes behave *semi-honestly*, a common assumption in secure distributed computation. This assumes nodes follow their protocol properly, but might also use their records of intermediate computations in order to attempt to learn other parties' private information. Then, we show the security of our protocol.

Lemma 3. Let T be the number of the final update. Assume that all nodes behave semi-honestly. Then, for all i, node i learns only $\pi_i^{(T)}$ and nothing else after the execution of secure spectral ranking in the link-aware model.

The security proof should be stated following the standardized proof methodology of secure multi-party computation with polynomial simulators as shown in [4]; however, because of the limitation of the space, we briefly explain why this protocol does not reveal private information.

The messages exchanged across the graph in this protocol are:

- step 3(b): $c_{ji,\mathsf{pk}} (= \mathrm{Enc}_{\mathsf{pk}}(b_{ji} q_j^{(t-1)}))$,
- step 4: decryption shares of $\mathrm{Enc}_{\mathsf{pk}}(q_i^{(t)})$.

We assume that there exists a node which eavesdrops all these messages exchanged across the graph. Then, we show that the node cannot derive any information other than the final output from what the node observed from these messages.

If node i observes all these messages, node i cannot solely decrypt $c_{ji,\mathsf{pk}}$ because the threshold cyrptosystem is applied. In step 4, node i can decrypt $\mathrm{Enc}_{\mathsf{pk}}(q_i^t)$ by gathering decryption shares, but this is the final output of the node i and does not violate the privacy. If a node other than node i observes all these messages, it cannot decrypt both $c_{ji,\mathsf{pk}}$ and the decryption shares. As shown, even when a node observes all exchanged messages, no knowledge other than the final output for the node can be learned.

Theorem 1. Assuming all nodes behave semi-honestly, secure spectral ranking correctly and securely computes the stationary distribution of link-aware private weighted graph in the sense of Statement 1.

The proof is readily derived from Lemma 2 and Lemma 3.

5.3 Expansion to PageRank: PrivateRank

Expansion of secure spectral ranking to PageRank, called PrivateRank, is straightforward. The transition probability from node i to node j is obtained as

$$p_{ij} \leftarrow (1-\epsilon)\frac{a_{ij}}{d_i} + \epsilon\frac{1}{n}. \tag{16}$$

Again, b_{ij} needs to be magnified by L as $b_{ij} \leftarrow Lp_{ij}$ such that $Lp_{ij} \in \mathbb{Z}_N$ for all i, j and $L\epsilon/n \in \mathbb{Z}_N$. Consider the power iteration with $B = (b_{ij})$. Given a vector of $(t-1)$th update $\boldsymbol{q}^{(t-1)}$, the tth update of the ith element is performed as

$$\begin{aligned} q_i^{(t)} &= \sum_{j=1}^{n} q_j^{(t)} b_{ji} = \sum_{j\in N_{in}(i)} q_j^{(t)} b_{ji} + \sum_{j\notin N_{in}(i)} q_j^{(t)} \frac{L\epsilon}{n} \\ &= \sum_{j\in N_{in}(i)} q_j^{(t)} b_{ji} + \frac{L\epsilon}{n}\left(LK^{t-1} - \sum_{j\in N_{in}(i)} q_j^{(t)}\right) \end{aligned} \tag{17}$$

Here we used the fact that $\sum_{i=1}^{n} q_j^{(t)} = LK^{t-1}$ again. By encrypting both sides of eq. 17 with pk, we obtain

$$\begin{aligned} \mathrm{Enc}_{\mathsf{pk}}(q_i^{(t)}) &\leftarrow \mathrm{Enc}_{\mathsf{pk}}\left(\sum_{j\in N_{in}(i)} q_j^{(t)} b_{ji} + \frac{L\epsilon}{n}\left(LK^{t-1} - \sum_{j\in N_{in}(i)} q_j^{(t)}\right)\right) \\ &= \left\{\prod_{j\in N_{in}(i)} \mathrm{Enc}_{\mathsf{pk}}(q_j^{(t-1)})^{b_{ji}}\right\} \cdot \left[\mathrm{Enc}_{\mathsf{pk}}(LK^{t-1}) \prod_{j\in N_{in}(i)} \left\{\mathrm{Enc}_{\mathsf{pk}}(q_j^{(t-1)})^{-1}\right\}\right]^{L\epsilon/n} \end{aligned} \tag{18}$$

Note that eq. 18 can be updated only with L, K, t, ϵ, n and $\mathrm{Enc}_{\mathsf{pk}}(q_j^{t-1})$ for $j \in N_{in}(i)$; these are possessed by node i. In PrivateRank, B is not a sparse matrix; however, by updating with eq. 18, this computation can be executed in $O(\Delta)$. By modifying the protocol of secure spectral ranking as follows, the protocol of PrivateRank is obtained:

- in step 1, p_{ij} is initialized with eq. 16
- in step 3(b), eq. 18 is used to update $\mathrm{Enc}_{\mathsf{pk}}(q_i^{(t)})$

The remaining steps are processed similarly. The security proof is omitted here.

5.4 Expansion to HITS: PrivateHITS

Expansion of secure spectral ranking to HITS, called PrivateHITS, is straightforward, too. In case of HITS, we assume that the ith node is responsible for the update of the value of hub and the value of authority of the node. Then, the computation of eq. 6 in the ith node are rewritten as follows:

$$a_i^{(t)} \leftarrow \sum_{j\in N_{in}(i)} h_j^{(t-1)}, \quad h_i^{(t)} \leftarrow \sum_{j\in N_{out}(i)} a_j^{(t)} \tag{19}$$

By encrypting both sides of these update equations,

$$\mathrm{Enc}_{\mathsf{pk}}(a_i^{(t)}) \leftarrow \mathrm{Enc}_{\mathsf{pk}}\left(\sum_{j\in N_{in}(i)} h_j^{(t-1)}\right) = \prod_{j\in N_{in}(i)} \mathrm{Enc}_{\mathsf{pk}}(h_j^{(t-1)}) \tag{20}$$

$$\mathrm{Enc}_{\mathsf{pk}}(h_i^{(t)}) \leftarrow \mathrm{Enc}_{\mathsf{pk}}\left(\sum_{j\in N_{out}(i)} a_j^{(t)}\right) = \prod_{j\in N_{in}(i)} \mathrm{Enc}_{\mathsf{pk}}(a_j^{(t)}) \tag{21}$$

are obtained. The updates are performed as follows. First, node i receives (resp. sends) $\mathrm{Enc}_{\mathsf{pk}}(h_j^{(t-1)})$ (resp. $\mathrm{Enc}_{\mathsf{pk}}(h_i^{(t-1)})$) from $j \in N_{in}(i)$ (resp. to $j \in N_{out}(i)$) . Then, updates $\mathrm{Enc}_{\mathsf{pk}}(h_i^{(t)})$ with eq. 20. Eq. 21 is updated at each node similarly. The security proof is omitted here, too.

5.5 Expansion to Collusion-resistant Protocol

In the beginning of this section, we assumed that all nodes use a common key set. In such a system, if more than t nodes collude together, the privacy of any node is compromised. We present the way to localize the coverage of the public key where each node independently generates a key set; the key set is valid only among neighbors of the node that generates the public key.

Let $N(i) = \{N_{in}(i), N_{out}(i)\}$ be the neighbor of node i, and let $\mathcal{K}(i) = \{\mathsf{pk}^i, \mathsf{sk}_0^i, \mathsf{sk}_1^i, ..., \mathsf{sk}_{|N(i)|}^i\}$ be a key set for node i. pk^i is commonly known by $\{i, N(i)\}$; sk_0^i is known only by i; the remaining private keys are privately distributed to each node in $N(i)$.

In this setting, values possessed by node $j \in N_{in}(i)$ are encrypted by public key pk^j of $\mathcal{K}(j)$. So, when node i computes the update in step 3(b) with a message sent from j, the encryption key of the message needs to be switched from pk^j to pk^i. Since node i and node j are neighbors each other, node i and node j hold one of secret keys of $\mathcal{K}(j)$ and $\mathcal{K}(i)$, respectively. When node j holds $\mathrm{Enc}_{\mathsf{pk}^j}(m)$, the protocol described below allows node i to learn $\mathrm{Enc}_{\mathsf{pk}^i}(m)$ with switching the public key securely.

1. node j: Generates $a \in_r \mathbb{Z}_N$. Then, send $c \leftarrow \mathrm{Enc}_{\mathsf{pk}^j}(m) \cdot \mathrm{Enc}_{\mathsf{pk}^j}(a)^{-1}$, $d_j \leftarrow \mathrm{Dec}_{\mathsf{sk}_0^j}(c)$, and $\mathrm{Enc}_{\mathsf{pk}^i}(a)$ to node i
2. node i: Computes $d_i \leftarrow \mathrm{Dec}_{\mathsf{sk}_i^j}(c)$ and recovers the messages as $m-a \leftarrow \mathrm{Rec}(d_i, d_j)$. Then, output $\mathrm{Enc}_{pk^i}(m) \leftarrow \mathrm{Enc}_{pk^i}(m-a) \cdot \mathrm{Enc}_{pk^i}(a)$

For simplicity, we describe the protocol for $(2,2)$-threshold cryptosystem; however, expansion for (m,t)-threshold cryptosystem is straightforward.

In step 3(b) of secure spectral ranking, the public key is switched to the public key of the node who receives by using this protocol. In this setting, even when more than t nodes collude together, node i's privacy is not compromised unless the number of colluding nodes in $N(i)$ is greater than t. In other words, node i can protect its privacy by itself by refusing suspicious nodes to join the protocol in the key distribution phase of node i.

5.6 Protocol for Link-unaware Model

In the link-aware case, node $j \in N_{in}(i)$ can directly send messages to node i without violating the privacy model. However, in the link-unaware model, establishing the connection in this protocol itself violates the link-unawareness as mentioned. If node i can receive messages from node $j \in N(i)$ with preserving the sender anonymity, the secure spectral ranking for the link-aware model is readily expanded to that for the link-unaware model.

For this, node $j \in N(i)$ uses the onion routing to send messages to node i in step 3(a). In step 3(b), node i can compute $\mathrm{Enc}_{\mathsf{pk}}(q_i^{(t)})$ by multiplying all messages sent from anonymous nodes. Unfortunately, even if the onion routing is used for the message delivery, the protocol is not secure in the sense of Statement 1 in the link-unaware model. Node i cannot know whose links are pointing to node i, but can know how many links are pointing to node i, by counting the number of messages anonymously transferred.

Since the onion routing supports real-time and bidirectional connections, the key-switching protocol can work via the onion routing. It follows that the localization of the coverage of public-keys are valid in the link-unaware model, too.

5.7 Discussion

External party. Although our protocol does not need any trusted third parties, some special nodes might be needed for the management of the overall communication, such as routing or broadcast, depending on the network design. Since this does not affect the security and the correctness of the protocol, we do not pursue this in this paper any further.

Exploitation of results. After the completion of the protocol, each node can disclose the final output in public if there is no risk for privacy violation. Even if the outputs cannot be disclosed, some knowledge, such as ranking of neighboring nodes, can be learned by introducing a security protocol implemented using SFE designed for the purpose.

Stronger security. Although our protocol guarantees the security under the semi-honest assumption, this does not necessarily mean that each node inputs their private link information correctly and follows the protocol description honestly. By making use of the techniques of zero-knowledge proofs, we can enforce each node to behave honestly to the specified protocol. To guarantee the correctness of the input, the protocol needs to be designed such that honest inputs itself benefit nodes (incentive-compatibility). Protocols meeting these conditions are remained for the future work.

6. EXPERIMENTAL ANALYSIS

We performed experiments to examine the efficiency of our protocols. Programs were implemented in Java 1.6.0. As the cryptosystem, [1] with 1024-bit keys was used. For SFE, Fairplay [8] was used. Experiments were carried out under Linux with Xeon 2.8GHz (CPU), 2GB (RAM).

Our solutions, PrivateRank or PrivateHITS, can learn exactly the same output as PageRank or HITS which do not consider privacy preservation. Therefore, our attention in these experiments is not on what kind of knowledge can be extracted from link structures, but on the computational efficiency of our solutions, and also on what kind of information is disclosed from the protocol execution. For the same reason, we used artificially generated random graphs for our experiments, in that the degree of each node is equally set to given Δ. Experiments with a large number of nodes are unrealistic. Instead, we simulated the network on a single computer; the computational time was measured without including the communication time. All results are averaged over 20 trials.

6.1 Link-aware Model

We compared the computational efficiency in PrivateRank (PR) in the link-aware model with that of both secure function evaluation (SFE) and decentralized spectral analysis (DSA). Since the computational complexity of a single update of these solutions is dependent only on the maximum degree Δ, Figure 2 shows the change in computational time per single update with respect to Δ. Table 2 shows the summary of the computation time and the disclosed information from the protocol execution.

DSA has been designed for spectral analysis on peer-to-peer networks; the computation is distributed equally among the nodes. The computational complexity per single update is $O(\Delta)$. Since DSA does not include any cryptographic operation, the computation time is quite small. DSA does not reveal the weights in the exchanged messages, so the privacy is preserved to some extent; however, the weights may be inferred from observed messages in some cases. Thus, DSA does not have theoretical/statistical guarantees of the privacy preservation.

SFE can originally achieve perfect privacy preservation as described in Statement 1; however we relaxed the requirement for the security in these experiments for the following reason. Although the computation time of SFE is bounded by the polynomial of the input size, if we strictly implement power iteration using SFE, the computation time would be unrealistically large. Instead, we implemented the SFE such that only a single update (step 3(a) and 3(b) of power iteration) is evaluated and then $x_i^{(t)}$ is returned to node i in each update. This allows the node to know intermediate information. Nevertheless, we can interpret the results of this version of SFE as the lower bound of the computation time of the strict implementation. As shown in Figure 2, the computation time of SFE is rather larger than that of the other solutions, even with the relaxed security.

We evaluated our PrivateRank in two settings, the common key setting and the collusion-resistant setting. PrivateRank can be considered a cryptographic extension of DSA. Therefore, the computation is equally distributed among the nodes and the computational complexity per single update at each node is $O(\Delta)$ as is that of DSA; however, the computation time is rather larger than DSA due to the cryptographic operations including SFE steps for normalization and convergence detection. The computation time of PrivateRank seems to be constant in $\Delta = 2^0$ and 2^4 in the figure. This is because the computation time for normalization dominates that of power iteration steps. In summary, in the link-aware model, we conclude that both efficient computation and perfect privacy preservation are achieved by our solutions.

6.2 Link-unaware Model

In next experiments, we compared our link-unaware PrivateHITS (PH) with privacy-preserving link analysis (PPLA). For the execution of PPLA, a semi-honest party called a tallier is required. At each update of PPLA, node i sends the encryption of $h_i^{(t)} a_{ij}$ for all $j \in V$ to the tallier; the tallier sums them for all i. Since the summation is centralized at the tallier, the computational complexity at each update is $O(n^2)$. Furthermore, PPLA allows node i to learn $a_i^{(t)}$ in the middle, so Statement 1 is not achievable.

Table 2: Comparison of the computation time and the information disclosed to node i in the link-aware model, $\Delta = 64$

	computation time (msec)	information disclosed to i
DSA	0.0093	$x_j^{(t)} p_{ji}$ for all t, j
PR (common key set)	592	none
PR (collusion-resistant)	1.97×10^4	none
SFE (relaxed)	8.14×10^4	$x_i^{(t)}$ for all t
SFE (strict)	$> 8.14 \times 10^4$	none

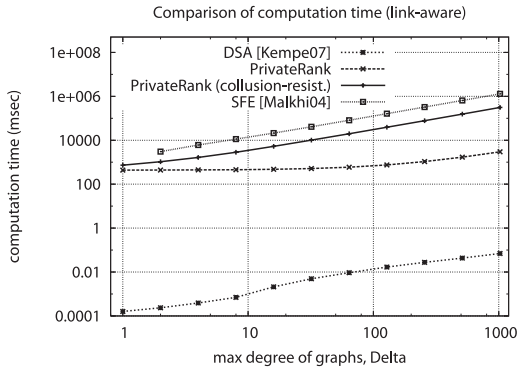

Figure 2: Max degree v.s. computational time in the link-aware model.

Table 3: Comparison of the computation time and the information disclosed to node i in the link-unaware model, $n = 8192$

	computation time (msec)	information disclosed to i
PH ($\Delta = 8$)	4718	$\lvert N_{in}(i)\rvert$
PH ($\Delta = 64$)	6379	$\lvert N_{in}(i)\rvert$
PH ($\Delta = 512$)	7268	$\lvert N_{in}(i)\rvert$
PPLA	1.087×10^7	$a_i^{(t)}$ for all t

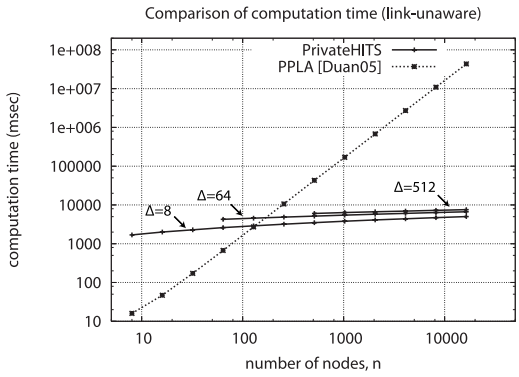

Figure 3: Number of nodes vs. computation time in the link-unaware model.

In our privateHITS in the link-unaware model, onion routing was used. The number of hops is dependent on the routing algorithm used; however, this can be at most $O(\log n)$ in general. Therefore, the computational complexity involved in setting up the encrypted layered structure of addresses is $O(\log n)$. In the routing process, each router strips off the outer layer and decrypts it in order to know the next destination. This process requires $O(\log n)$ decryption. The complexity of the remaining steps is the same as that of the link-aware model. Thus, the computational complexity of link-unaware PrivateHITS per single update at each node is $O(\Delta + \log n)$.

Figure 3 shows the change in computation time with respect to the number of the nodes per single update. The computation time of PPLA shows quadratic growth. If $n = 2^{13} (= 8192)$, the computation time of PPLA at each update is around three hours. The computation time of privateHITS is measured for $\Delta = 2^3, 2^7, 2^9$. If $n = 2^{13}$ and $\Delta = 2^7$, the computation time of PrivateHITS per single update is around six seconds. As shown, if the graph is sparse, our solution performs more efficiently. In the link-unaware model, although our solution allows node i to learn $|N_{in}(i)|$, intermediate messages are not revealed and the anonymity of the links is theoretically preserved.

7. CONCLUSION

In this paper, we formally stated the concept of link-aware/link-unaware private weighted graphs and investigate secure link analysis for private weighted graphs.

The concept of secure computations using private weighted graphs opens up a way of exploring knowledge from various privately connected entities. Graph mining problems, such as node clustering, link prediction, and frequent pattern extraction of private weighted graphs are promising future problems.

8. REFERENCES

[1] I. Dåmgard and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography 2001*. Springer, 2001.

[2] I. Dåmgard. and M. Koprowski. Practical threshold RSA signatures without a trusted dealer. In *Proceedings of EuroCrypt 2001*, Lecture Notes in Computer Science 2045, pages 152–165, 2001, Springer, 2001.

[3] Y. Duan, J. Wang, M. Kam, and J. Canny. Privacy Preserving Link Analysis on Dynamic Weighted Graph. *Computational & Mathematical Organization Theory*, 11(2):141–159, 2005.

[4] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[5] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[6] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences*, 74(1):70–83, 2008.

[7] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[8] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay: a secure two-party computation system. In *Proceedings of the 13th USENIX Security Symposium*, pages 287–302, 2004.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1998.

[10] J. Sakuma, S. Kobayashi, and R.N. Wright. Privacy-preserving reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 864–871. ACM New York, NY, USA, 2008.

[11] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.