Privacy-Preserving Indexing for eHealth Information Networks

Yuzhe Tang, Ting Wang, Ling Liu, Shicong Meng, and Balaji Palanisamy
College of Computing, Georgia Institute of Technology
Atlanta, GA 30332, USA
yztang@gatech.edu, {twang, lingliu, smeng, balaji}@cc.gatech.edu

ABSTRACT

The past few years have witnessed an increasing demand for the next generation health information networks (e.g., NHIN[1]), which hold the promise of supporting large-scale information sharing across a network formed by autonomous healthcare providers. One fundamental capability of such information network is to support efficient, privacy-preserving (for both users and providers) search over the distributed, access controlled healthcare documents. In this paper we focus on addressing the privacy concerns of content providers; that is, the search should not reveal the specific association between contents and providers (a.k.a. content privacy). We propose SS-PPI, a novel privacy-preserving index abstraction, which, in conjunction of distributed access control-enforced search protocols, provides theoretically guaranteed protection of content privacy. Compared with existing proposals (e.g., flipping privacy-preserving index[2]), our solution highlights with a series of distinct features: (a) it incorporates access control policies in the privacy-preserving index, which improves both search efficiency and attack resilience; (b) it employs a fast index construction protocol via a novel use of the secrete-sharing scheme in a fully distributed manner (without trusted third party), requiring only constant (typically two) round of communication; (c) it provides information-theoretic security against colluding adversaries during index construction as well as query answering. We conduct both formal analysis and experimental evaluation of SS-PPI and show that it outperforms the state-of-the-art solutions in terms of both privacy protection and execution efficiency.

Categories and Subject Descriptors: D.4.6 [Security and Protection]: Access controls H.3.1 [Content Analysis and Indexing]: Indexing methods H.3.3 [Information Search and Retrieval]: Search process

General Terms: Security Algorithm

Keywords: Privacy preserving protocol, keyword search, distributed indexing

1. INTRODUCTION

Many healthcare providers, payers, and pharmaceutical companies have increased their use of eHealth solutions to manage health-related information and to automate administrative and clinical functions. We are witnessing an increasing demand for the next generation health information networks, which hold the promise of providing large-scale information sharing over distributed, access controlled content across a network of healthcare providers. A representative example is the work currently under way to construct the Nationwide Health Information Network (NHIN) [1] that supports information sharing among more than 20 federal agencies, along with numerous private hospitals and doctors' offices. A fundamental capability of such information network is to provide privacy-preserving search over distributed, access controlled content.

More specifically, individual healthcare providers typically enforce strict regulations over the healthcare information for a number of reasons, such as patients' privacy requirements, conflicting economic interests, and federal administration; meanwhile, the capability of efficiently identifying and retrieving relevant content across healthcare administrative boundaries is crucial for the healthcare information network to improve care quality, emergency response, and diagnosis accuracy. This poses the question of how to facilitate effective search while minimally revealing which providers possess which content (i.e., content privacy).

A naive approach to achieve privacy-aware search is query broadcasting: each query is forwarded to all the participating providers; those providers with matched content may then contact the querier directly. Content privacy is best preserved in this manner, since no sensitive information is required to route the query. However, such global-scale probing is not scalable with respect to the number of providers, in terms of both communication bandwidth and query latency. In the case of selective queries that most providers do not have matched content, broadcasting results in huge waste of communication and computation resources.

Alternative is to use a centralized index server (in implementation which can be readily replicated) that holds an indexing structure to facilitate the query routing. Query efficiency and scalability is attained for each query is only re-directed to providers which are bound to have matched contents. However, to construct the indexing structure, which is publicly accessible, typically requires providers to fully expose their content possession information. In this sense, the centralized index server must be trusted by all participating providers in its behavior. In healthcare applications, however, not only the enormous trust is impractical for providers with conflicting interests, but also the centralized architecture is vulnerable to security attacks and suffers from single point of failure.

The first Privacy Preserving Index (PPI) was proposed in [2] to strike a balance between privacy preservation and search efficiency. We refer to this approach as flipping PPI. It leverages an abstraction of group-wise index, by which providers are organized into a set of disjoint *privacy groups*.[1] Content privacy is preserved in the sense that providers in the same group are indistinguishable. Query dissemination is performed at the granularity of privacy group. Within a privacy group, a given query is forwarded either to all providers or to none of them, depending on whether there exists (at least) one provider with matched content. Flipping PPI is known to suffer from index construction inefficiency, due to the large number of rounds of PPI computation, and is vulnerable to colluding attacks.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

[1]We use "privacy group" and "group" interchangeably when no confusion occurs.

In sum, we argue that existing privacy preserving index and search protocols suffer from the following drawbacks:

- **Ineffectiveness**: Flipping PPI is ineffective on making trade-off between privacy preservation and search efficiency. In the case that a considerable number of providers possess content matching the query, the flipping PPI approach essentially degrades to query broadcasting which as aforementioned suffers from scalability in terms of search performance. We elaborate on this in the start of Section 3.1.

- **Inefficient index construction**: Efficient construction of group-wise index is crucial for indexing frequently updated content. Existing schemes suffer from either complexity of computation by using secure multiparty computation [9], or excessive privacy leakage by using tailored algorithms. Flipping PPI [2] employs a randomized and iterative algorithm that is neither time-efficient nor privacy preserving. To build index, flipping PPI needs to run multiple rounds (typically above 20 rounds); in each round a sequential scan ($O(n)$ time complexity where n is group size) is required. Flipping PPI's index construction protocol also features weak privacy preservation, since it is vulnerable to colluding attacks with even two collaborators.

- **Role-insensitivity**: Existing PPI approaches are either role insensitive or preserve no privacy in terms of provider's access control policy. Typically, a fully trusted index server is required to conduct user authentication, which is not scalable or practical [23]. Queries are redirected to the same set of providers regardless of the role of the querier in the access control policy. We argue that this role insensitivity may lead to both query routing inefficiency and privacy leakage. Consider a querier having access rights to a very limited set of providers poses a query for a common term. PPI returns a large set of providers that possess this term, yet a handful of which are accessible to the querier, yielding unnecessary query overhead. In contrast, a querier is directed to a group from which she gets negative results from $n-1$ providers but does not have access to the remaining one. The querier can then ensure the last provider must hold the term in question, leading to privacy leakage.

In this paper, we propose the concept of *role-sensitive PPI* and a secure protocol for fast construction of PPI based on the primitives of secret sharing. We entitle this concrete implementation of our role-sensitive PPI as SS-PPI. Given a query in conjunction with the role of the querier, our *role-sensitive PPI* returns the groups of providers that potentially hold the content that matches the search term for this role. Both privacy preserving and search efficiency are attained on a finer-granularity level. Since our PPI abstraction is defined on fine granularity (role sensitive), we preserve content privacy as well as *access policy privacy* at the same time. This new type of privacy protects the sensitive information in provider-defined access rules, such as to which role provider p has granted access. To the best of our knowledge, this paper is the first one addressing the privacy of access policy in the PPI framework. Technically, our index construction protocol (SS-PPI) makes a novel use of secret sharing without requiring third party involved, which comparing to existing secret-sharing schemes sees better scalability. It is efficient in the sense that all "secret shares[2]" are constructed in a distributed, parallel manner with constant complexity of computation and communication. In particular, it finishes in 2 stages, each expected to take 1 time unit (for reasonable setting of group size n, like $1k$). More importantly, our SS-PPI protocol achieves information-theoretic security against colluding attacks. For less than $2c-2$ adversaries in the network, any provider's privacy is not leaked , where c is a system parameter.

[2]Packets and shares are used interchangeably in the following paper.

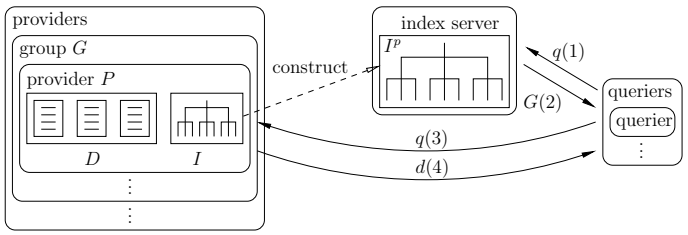


Figure 1: Multi-source information search and retrieval.

The remaining of this paper is organized as follows. Section 2 presents the overview of our system, and Section 3 elaborates privacy-aware index construction protocol. Security analysis is given in Section 4. Experimental results are shown in Section 5. Finally, Section 6 surveys related work and Section 7 concludes this paper.

2. OVERVIEW

In this section, we formally describe the system architecture, our role-sensitive PPI and then overview the secret-sharing based protocol for index construction.

2.1 Architecture of Role-sensitive PPI

We first give an overview of our targeted multi-source information sharing infrastructure, as shown in Figure 1. The infrastructure mainly involves three entities, including a set of autonomous information providers, an index server, and a set of queriers (or users who consume the information)[3].

Each provider holds at its depository a set of documents and has self-specified privacy requirements and regulations. The set of information providers jointly provide information search for users. And each provider itself is responsible for search processing (at a finer granularity) and data retrieval. This way, each provider is able to determine the disclosure of information to the corresponding querier at its discretion. Specifically, provider p possesses a depository of private documents, $D(p)$, on which a role-based access control policy is enforced. We assume that each document d can be described by a set of content descriptors (CD), denoted by $T(d)$ [6] (e.g., keywords), from a finite set of CDs T. Provider p can summarizes its local data and access policy by bitmap $b(p)$, where each column is associated with a role o and each row with a content descriptor or term t. The cell at column o and row t is a bit, with 1 meaning provider p has one or more documents matched to term t and accessible to subject role with o, and 0 otherwise. The bitmap $b(p)$ is intended for being published to the index server. Also, each provider optionally maintains a local indexing structure I regarding its documents to facilitate local search and access enforcement.

A querier searches information by issuing queries and presenting her roles. A query q is represented by a set of CDs, denoted by $T(q)$ and querier's roles $O(q)$. Document d satisfies query q if $T(q) \subseteq T(d)$ and d is accessible to certain role in $O(q)$. From the local bitmap's point of view, provider p can answer query q if for every row in $T(q)$, there are at least one cell that is 1 for all columns in $O(q)$. In the framework, a query is successfully answered if *all* the information providers that possess documents satisfying the query receive the query. It is at the discretion of the corresponding information providers to enforce access to the relevant documents (retrieval), which may involve further steps such as charge negotiation. In this paper, we focus on the phase of information search.

Overall, a query q can be answered in the following steps, as shown in Figure 1. The querier sends query q consisting of $T(q)$ and $O(q)$ to the index server, (step (1)), which by looking up the public privacy preserving index I_p returns the identity information of the corresponding providers G (step (2)). The querier then issues q to these providers (step (3)), who at their discretion contact the querier and provide access of documents d that satisfy q (step (4)).

[3]Following, we use "querier" and "user" interchangeably.

2.2 Privacy Goal in Role-sensitive PPI

In a typical PPI system, there are privacy of various information needed to be protected, including content privacy, *policy privacy* and query privacy, among many others. While query privacy, which involves with searcher's identity being kept from disclosure, can be protected by anonymity protocols (e.g. [20, 18, 17]), content privacy and policy privacy are the ones we primarily address in this paper. Following flipping PPI [2], we define content privacy as below,

Definition *Given provider p and term t, content privacy is defined to be information about provider p's possession of certain document containing sensitive term t. Content privacy is leaked when one party without access to provider p can claim with certainty if provider p possesses term t.*

In addition to content privacy, we identify the access policy privacy in generic PPI systems, since our published index is fine-grained and incorporates the access policy information in it. The formal definition is introduced as follows,

Definition *Given provider p and role o, access policy privacy is defined to be the information about p has granted access to o on certain document that provider p holds. Policy privacy is leaked when one party without access to privoder p can claim with certainty if provider p grants access to users with role o.*

In our architecture, a querier is expected to report her true role, but also allowed for falsely reported roles, namely the one she is not associated with. No authentication or access control are enforced, on our public, untrusted index server. Our role-sensitive PPI is well-designed that querier reporting true role obtains a list of providers that cover full set of query answers, and that querier falsely reporting her roles receives a somewhat meaningless list of providers from which she can't deduce any sensitive knowledge in terms of content privacy and policy privacy.

2.3 Index Construction Protocol

Motivated by the deficits of existing solutions in facing large-scale multi-source information sharing infrastructures, we propose SS-PPI, a novel indexing scheme that supports information network comprising thousands of content providers, and provides theoretical guarantees on both possession privacy protection and execution efficiency.

The framework of SS-PPI mainly consists of two phases, index construction and query answering. The former further processes in three major components, group formation, group aggregation and global index construction.

- *Grouping*. SS-PPI organizes providers into privacy groups. In this paper, we adopt the strategy of random grouping based on universal hashing [2].

- *Secure group aggregation*. Within each privacy group, a summarization structure is built in a privacy-preserving manner that indexes the content possession by group members. Instead of using the generic circuit computing [9], SS-PPI constructs this aggregation by an extended secrete sharing scheme [19], which endows our solution with both scalability and attack-resilience. Our analysis and experiments show that the scheme can achieve information-theoretic privacy of a provider even when multiple providers within the group are compromised.

- *Privacy-aware global index construction*. The global index is constructed efficiently by merging the set of group aggregations. SS-PPI adopts a distributed scheme that not only amortizes the trust on a single third party, but also supports localized, incremental index update.

3. PRIVACY-PRESERVING INDEX CONSTRUCTION

In this section, we present in detail the phase of privacy-preserving index construction in SS-PPI. It entails three main components; group formation, secure group aggregation, and global index construction.

3.1 Random Group Formation

Group formation is the process that organizes providers into different privacy groups. In the design of a good formation strategy that strike a balance between search efficiency and privacy preserving, group term selectivity is critical. Group term selectivity refers to the ratios of providers in a group that possesses the term to the group size. The less group-wise selective a term is, the harder an adversary can pinpoint a provider that possesses a specific term, thus better privacy preserving. On the other hand, less group term selectivity implies queries are forwarded to more providers with no answer, thus more bandwidth overheads. For a given group size, the goal of group formation is to make all indexed terms reach an expected values in group selectivity.

We follow in this paper the conventional approach, that is, random grouping (e.g. [2]), which randomly assigns providers to groups. Group size is a critical factor to utility and privacy preserving of published index. When group size is configured to be too big, it could easily make the group-wise index degrade to query broadcasting. Consider a group of size n and a term with global selectivity q, the probability for a group to be negative (i.e., no providers in it having the term) is $(1-q)^n$. This value quickly approaches 0 as n grows, regardless of value of q. For example, when $q = 0.5, n = 10$, the value is 0.1%, and when $n = 20$, the negative probability becomes 10^{-6}, implying all groups are positive ones and query broadcasting is thus required. On the other hand, when group size n is configured to be too small, privacy could easily be leaked since the possibility for providers in one group to all be positive is q^n which is fairly large for small n. As shown in experiment part, we empirically set the value of group size and show its effectiveness.

3.2 Secure Group Aggregation: Design Rationale

After the privacy groups are formed, one needs to aggregate the group-wise term-possession index. The technical goal is to protect individuals' privacy during the process in which group-wise index is formed. That is, one can not guess with confidence higher than what final aggregated index discloses. In particular, we address two technical goals,

- We firstly address privacy preserving in the presence of colluding attacks. Our privacy goal can be guaranteed, if the total number of colluders is bounded below a certain threshold.

- We secondly address performance in terms of both bandwidth and time. Our designs aims at best performance as long as privacy is not sacrificed.

With regards to the above goals, existing protocols show flaws in a way or two. One conventional scheme is flipping PPI's iterative randomized algorithm. However, it is time-inefficient; it requires to run multiple iterations before the final group index reaches certain level of accuracy. This problem is compounded when group-wise index is frequently updated, resulting in prohibitively high cost for re-constructing the indexing structure. More importantly, its publishing process is vulnerable to colluding attacks, specially in the case of an innocent provider ending up with its predecessor and successor both being malicious. Another possible technique can be used is the generic secure multi-party computation [6] which consumes considerable computation overheads.

3.3 Secure Group Aggregation: The Algorithms

Inspired by the observation above, we propose a secret-sharing based scheme for fast and secure index construction. Our novel secure group aggregation scheme is based on an extended secrete sharing protocol [19], which achieves constant communication cost in group aggregation and provides strong privacy protection. Within each group, group aggregation collects the statistics of the possession of providers in the group with respect to each query term and access role. We assume that the providers have already been grouped and placed into in group-wise overlay. Without loss of generality, we consider a specific group comprising a set of providers $p_0, p_1, \ldots, p_{n-1}$, each holding a private value $v_i \in \{0, 1\} = \mathbb{Z}_2$, called *sub-secret*, corresponding to a specific term and access role[4]. The output, called *super-value* v, is the number of providers with $v_i = 1$, that is,

$$v = \sum_{i=0}^{n-1} v_i \tag{1}$$

where the super-value v can span from 0 to n. Our protocol computes the super-value accurately and securely. Each involved provider works in four stages: generates sub-packets for sub-secret, distributes sub-packets, computes super-packets from received sub-packets, and aggregates the super-packets to form super-value. Next, we discuss the four stages in details.

3.3.1 Generating sub-packets

In this stage, each provider p_i splits its sub-secret v_i into c sub-packets $u_{i,j}$, such that their modulo sum equals v_i, formally,

$$v_i = f(u_{i,0}, u_{i,1}, \ldots, u_{i,c-1}) = \sum_{j=0}^{c-1} u_{i,j} \mod q \tag{2}$$

where q is the modulus with $q \gg n$, and each sub-packet $u_{i,j}$ is defined on the packet domain $\mathbb{Z}_q$. The packet-generating process generates a (c, c)-secret packets; that is, given any less than c sub-packets, the sub-secret v_i is still completely undeterminable. A set of implementations are available to generate the sub-packets[5]. In this paper, we select the simplest one: randomly and independently pick the first $c-1$ sub-packets $u_{i,j}$'s (for $j = \{0, \ldots, c-2\}$), and let the last one be $u_{i,c} = (v_i - \sum_{j=1}^{c-1} u_{i,j}) \mod q$. In Appendix, we prove this is a (c, c)-secret sharing in Theorem A.1.

3.3.2 Distributing sub-packets

We assume that the providers in a group are structured into a ring-like overlay, as illustrated in Fig. 2. In the overlay, each provider p_i has $c-1$ neighbors in the clockwise direction, $p_{h(i,1)}, \ldots, p_{h(i,j)}, \ldots, p_{h(i,c-1)}$, where $h(i, j)$ denotes the index of the j-th neighbor of p_i. A bi-directional secure channel is set up on each neighboring connection. Provider p_i also has $c-1$ neighbors in the counter-clockwise direction, $p_{h'(i,1)}, \ldots, p_{h'(i,j)}, \ldots, p_{h'(i,c-1)}$, where $h'(i, j)$ denotes the index of the provider whose j-th clockwise neighbor is p_i, that is, $h(h'(i, j), j) = i$. In total, each provider knows $2c-2$ neighbors in the group, and unaware of the rest's positions. In particular, the 0-th neighbor of p_i is itself, i.e., $h(i, 0) = i$. A variety of instantiations of $h(\cdot, \cdot)$ are possible. At this point, for simplicity, we assume $h_1(i, j) = i + j \mod n$, that is, p_i take the nearest-$(c-1)$ providers to be its neighbors. For instance, in Fig. 2, p_0 knows p_1, p_2, p_6, p_7 and no more; p_0 knows p_6 because mutual communications are required when p_6 is to set up a secure channel with p_0.

Provider p_i proceeds to distributing the sub-packets to its neighbors: $u_{i,j}$ will be sent to $p_{h(i,j)}$. In particular, since $h(i, 0) = i$, the first sub-packet $u_{i,0}$ is always kept locally on p_i. During this stage, all communication are through the secure channels; that is, they are encrypted and authenticated. Messages from different providers are sent asynchronously and in parallel, and thus consuming $O(1)$ time-unit.

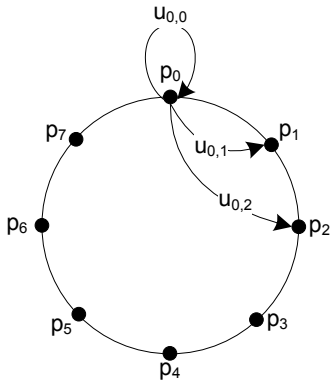


Figure 2: Overlay of secret packet distribution (with $c = 3, n = 8$)

3.3.3 Computing super-packets

While sending out sub-packets, providers also receive sub-packets from other providers. Specifically, p_i receives from its previous neighbor p_{i-j} sub-packet $u_{i-j,j}$, where $1 \le j \le c-1$. p_i then sum them up, together with its local sub-packet $u_{i,0}$, to get the super-packet u_i,

$$u_i = \sum_{j=0}^{c-1} u_{i-j,j} \mod q \tag{3}$$

Thus, $u_{i,0} = (u_i - \sum_{j=1}^{c-1} u_{i,j}) \mod q$. Combining it with Equation 2, we get,

$$\begin{aligned} v_i &= (u_i - \sum_{j=1}^{c-1} u_{i-j,j} + \sum_{j=1}^{c-1} u_{i,j}) \mod q \\ &= f'(u_i, u_{i-c+1,c-1}, \ldots, u_{i-1,1}, u_{i,1}, \ldots, u_{i,c-2}, u_{i,c-1}) \end{aligned} \tag{4}$$

Interestingly, Equation 4 defines a $(2c-1, 2c-1)$ secret sharing scheme, as will be proved in Theorem 4.1. It implies that even with $2c-2$ packets, the secret value v_i is still completely undeterminable. Note that each input sub-packet in Equation 4 corresponds to a distinct remote provider, e.g., $u_{i-c+1,c-1}$ is received from p_{i-c+1}, u_{i+1} is sent to p_{i+1}, and so on. When the group size n is large, e.g., containing at least $2c-1$ or $2c-2$ providers (depending on how u_i is aggregated, as below), our protocol is resilient to collusion attacks of up to $2c-2$ or $2c-3$ malicious providers in the group.

3.3.4 Aggregating super-packets

In the last stage, super-packets from all providers are aggregated to form super-value $u = (u_0 + u_1 + \cdots + u_{n-1}) \mod q$. In our implementation, we adopt the most time-efficient one: every provider p_i sends u_i to a special provider, say p_0 which is then responsible for summing up all u_i's. The final results u will be sent to index server as part of the privacy preserving index. The whole process is sketched in Algorithm 1.

[4] As will be discussed later, the aggregation process of v_i for different terms and roles can be combined together and related messages can be piggybacked.

[5] For example, Shamir's secret sharing[19] is a possible way to generate $u_{i,j}$ in the form of Equation 2. Specifically, for sub-secret v_i, we randomly generate a polynomial $g_i(x)$, s.t. $g_i(0) = v_i$. Also there are c input values x_j's ($j \in \{0, 1, \ldots, c-1\}$) that are globally-agreed on. Applying x_j on g_i, we have $y_{i,j} = g_i(x_j)$. Applying Lagrange Interpolation, $v_i = g_i(0) = \sum_{j=0}^{c-1} \mathbb{L}_{i,j}(0) \cdot y_{i,j}$. By picking $u_{i,j} = \mathbb{L}_{i,j}(0) \cdot y_{i,j}$, we get equation 2 (without modulo operation, though).

Algorithm 1 index-construction(provider p_i, sub-secret v_i)

1: {Generating sub-packets}
2: **for all** $j \in [0, c-2]$ **do**
3: $u_{i,j} \leftarrow$ random number in $\mathbb{Z}_q$
4: $u_{i,c-1} \leftarrow (v_i - \sum_{j=1}^{c-1} u_{i,j}) \mod q$
5: {Distributing sub-packets}
6: **for all** $j \in [1, c-1]$ **do**
7: p_i sends $u_{i,j}$ to provider p_{i+j}
8: {Computing super-packets}
9: **for all** $j \in [1, c-1]$ **do**
10: p_i receives $u_{i-j,j}$ from provider p_{i-j}
11: $u_i \leftarrow (u_{i-c+1,c-1} + u_{i-c+2,c-2} + \cdots + u_{i-1,1} + u_{i,0}) \mod q$
12: {Aggregating super-packets}
13: p_i send u_i to p_0 (p_0 will then sum up all received super-packets)

Example. Consider four providers with security parameter $q = 5, c = 3$. In the first round, every provider p_i starts with splitting her sub-secret v_i into $c = 3$ sub-packets. For instance, on p_1, $v_1 = 0 = (2 + 3 + 0) \mod 5$, where the first two numbers are randomly distributed in domain $\mathbb{Z}_q = \{0, 1, 2, 3, 4\}$. Then, p_1 sends out the sub-packets, e.g., 0 to p_3, 3 to p_2, but 2 kept locally. At the same time, p_1 receives sub-packets from other providers, e.g., 0 from p_3 and 1 from p_4. Then p_1 sums up all these sub-packets to get the super-packet $u_1 = (0 + 1 + 2) \mod 5 = 3$. Other providers work in a similar manner, and they further send all super-packets to p_1. At the end, p_1 sums up all super-packets $(3 + 0 + 2 + 2) \mod 5 = 2$.

(q=5,c=3)	p_1	p_2	p_3	p_4
v_i	0	1	1	0
$u_{i,3}$	0	4	0	3
$u_{i,2}$	3	3	2	1
$u_{i,1}$	2	4	4	1
$u_{i-2,3}$	0	3	0	4
$u_{i-1,2}$	1	3	3	2
$u_{i,1}$	2	4	4	1
$u_i=\sum u_{i-j+1,j}$	3	0	2	2
v	2			

Figure 3: Secret Sharing-based Index Construction Example

3.4 Analysis of Protocol Correctness and Complexity

Our protocol outputs the correct value of v, namely, $u = v$, as below,

$$\begin{aligned} u &= \sum_{l=0}^{n-1} u_l \mod q = (\sum_{l=0}^{n-1}\sum_{j=0}^{c-1} u_{l-j,j} \mod q) \mod q \\ &= (\sum_{j=0}^{c-1}\sum_{l=0}^{n-1} u_{l-j,j}) \mod q = (\sum_{j=0}^{c-1}\sum_{i=-j}^{n-1-j} u_{i,j}) \mod q \\ &= (\sum_{j=0}^{c-1}\sum_{i=0}^{n-1} u_{i,j}) mod q = (\sum_{i=0}^{n-1}\sum_{j=0}^{c-1} u_{i,j} \mod q) \mod q \\ &= \sum_{i=0}^{n-1} v_i \mod q = \sum_{i=0}^{n-1} v_i = v \end{aligned} \quad (5)$$

The last step is due to that $q \gg n$, and $v \in \{0, 1, \ldots, n\}$. Hence, in the final public index, for each provider having $v_i = 1$, the group will have aggregated sum $v \geq 1$, thus being present in the posting list. Completeness can be attained. Besides, we use posting-list intersection for multi-keyword query processing, thus query consistency is achieved; that is, the results of multiple-keyword query equals the intersection of all results of multiple single-keyword queries. In short, our protocol is correct.

Efficiency of our protocol can be analyzed in two aspects: the time latency and bandwidth overhead. Stage 1 of our protocol requires $n \cdot c$ messages to be sent, and can finish within O(1) time-unit due to parallelism. The actual latency depends on the slowest provider in the network. In this stage, the bandwidth overhead also implies the costs of setting up secure channels. For second stage, the basic aggregation protocol can finish within $O(1)$ time and with n messages.

Overall, the above protocol runs for one single term and one access role. To build PPI for multiple terms and access roles, it runs multiple single-term protocols, independently with each other. Specifically, the aggregation process carries a set of bitmaps or vectors, each vector corresponding to a role and each vector bit summarizing the possession information regarding to an indexed term. Sending and merging different bits or vectors can be piggybacked in the same messages and packets the same secure channel. By this way, time latency stays unchanged with growing number of indexed terms and roles.

3.5 Global Index Construction

The group-wise indices are not ready for direct use by the index server to route queries, since they may, to some extent, violate the content privacy of participating providers. It happens when a majority number of providers in a group hold a (set of) specific term(s), which makes it possible for the adversary to identify the content possession of these providers with high confidence. Therefore, we need enforce another layer of protection before the global index is published. We achieve this by adjusting and merging group-wise indices to quantitatively meet providers' privacy needs. In the following, we will introduce a group merging process. This process is optional in our system, because it may requires a partially trusted coordinator. However, after merging, privacy preserving can attain certain quality as system/providers want. It is noteworthy here that security of the coordinator can be strengthened by making it periodically offline [13].

3.5.1 Group-wise Index Adjustment

The goal of our adjusting/merging process is to guarantee each group g has positive providers with respect to each term lower than a threshold $q \cdot n(g)$, where $n(g)$ is group size and q is a predefined privacy parameter. We consider a centralized coordinator which collects the group-wise indices and merges them to address the participating providers' privacy in a best-effort manner. For example, for term t, coordinator will find the group with the most positive providers g and the one with the least positive providers g'[6]. If g contains more positive providers than the threshold, the coordinator merges g with g' such that the percentage of positive providers $\frac{v(t,g)+v(t,g')}{n(g)+n(g')}$ drops in the merged group, where $v(t, g)$ denotes the percentage of positive providers regarding to term t. If the merged percentage is still too big, that is, bigger than q, the merging process continues to merge with the group currently with smallest $\frac{v(t,g)}{n(g)}$. The whole process stops when the merged percentage drops below q.[7]. For multiple terms, we define a metric $\alpha(g)$ for group g,

$$\alpha(g) = \sum_{t \in ST} v(t, g) - q \cdot n(g) \quad (6)$$

[6]We address the global meta-information, like the number of positive providers, can be leaked since it does not disclose any information on which provider is positive.

[7]The stop condition must be met in certain iteration, for we assume the threshold q is bigger than global term selectivity. It is a reasonable restriction, since otherwise even query broadcasting can't make all group-wise percentage of positive providers be smaller than term selectivity.

where ST is the subset of terms that are privacy-sensitive, defined by providers. For multiple terms, in each iteration, we try to merge group g with lowest $\alpha(g)$, and the rest similarly runs. In practice, in case that there is no trusted coordinator, one can omit this process and privacy preserving is attained in a best effort way.

3.5.2 Global Rank-aware Index Construction

The group-wise index of each group is combined into a single global index. A naïve way is to let each group send her index to the centralized index server and to combine locally there. In this scheme, the percentage of positive providers in each group is then known by the index server, which however is sensitive information. In order to avoid this privacy breach, we adopt the privacy-preserving protocol [22] to output posting index ordered by positive percentage but without publishing the percentages. To further save the query processing costs, one can only publish Top-K groups with most positive providers, where K is a system parameter. Depending on the value of K, this strategy makes a trade-off between query performance and result recall. For example, small K can result in more efficient query processing, but not more effective. That is, query recall may be harmed, since certain results appear in bottom of the ordered posting lists. In practice, K should be set according to users' demands on result quality and system resource budget.

3.5.3 Index Updates

In real systems, data may be frequently updated and new providers may come and leave. A naive way to handle this update is to rebuild our privacy preserving index from scratch every time an update occurs. For index update efficiency, we adopt an incremental, batch-oriented update approach. Whenever there are t providers that newly join or locally updates their index, we re-run the index construction protocol among the updated providers. Thus, updates can not be tied to any specific provider out of t. In addition, since we add noise and do not disclose exact value of term selectivity (as discussed in Section 4.2.4), content privacy in update is preserved.

4. PROTOCOL SECURITY ANALYSIS

In our system, privacy leakage consists of two parts, one $P1$ is during index construction and the other $P2$ after index gets published. For $P1$, in our protocol for index construction, there are four kinds of information; the "super-value" v, the "super-packets" u_l, the "sub-packets" $u_{i,j}$ and "sub-secret" v_i. The super-value v is the final result and is made publicly known. Out of the four, only "sub-secret" v_i's is the private information to protect.

4.1 Privacy Model

4.1.1 Privacy Adversaries

For security analysis, we start with adversaries of different roles with different capabilities. Adversaries could be providers, network eavesdroppers, searchers and even the index server. However, to make a system function correctly, one needs to assume minimal level of trusts. For instance, providers and the index server in our network are assumed to be semi-honest, implying they follow our protocol specification, but may attempt to learn additional information by analyzing the transcript of messages received during the execution[15]. More specifically, providers will follow the secret-sharing based index construction protocol and index server will perform work for query answering, like posting list intersections. A network eavesdropper could passively log the messages under her surveillance. She can have the global power to monitor all messages coming through the network or the local power to monitor messages sent out by a particular providers. A searcher could pose queries to the index server in the wish to obtain sensitive information. Overall, an adversary in our model can assume multiple roles, like she could be one of participating providers and can pose queries to index server at the same time. Or multiple adversaries can collude to gain more knowledge.

For privacy $P1$, providers can obtain more information than other roles, since all messages are encrypted in our network and only providers can see the content/payload of the network messages.

4.1.2 Privacy Metric

Recall that content privacy and policy privacy are addressed in this paper. The degree of privacy preservation is quantitatively measured by the probability with which an adversary's claim on sensitive knowledge fails. The actual claim differs for different types of privacy, as defined above. The probability is equal to the percentage of false positive providers in the result list, which is used as privacy metric in our system. For instance, in the result list of 10 providers, if they all possess a sensitive term t, content privacy is definitely leaked. Because for any provider p in the list, adversary can claim p possess term t in question and such claim is true with 100% probability. On the other hand, when there are 5 false positive providers in the list, in the case that adversary randomly pick a provider to perform attack, the probability for her claim to fail is at least half (50%). In this paper, we choose the "Probable Innocence"[2, 18]) as our main quantitative privacy goal, in which the false positive rate should be higher than one half 0.5.

4.1.3 Attack Model

Our general attack model involves that a security role (e.g., a provider) observing certain messages from other parties makes claim on sensitive knowledge that breaches privacy. We consider attack to breach both privacy $P1$ and privacy $P2$. 1) For privacy $P1$, we consider both attacker as a single provider and attackers as colluding providers. We assume that a single provider observing the messages from its neighbors always claim its neighbor has the term (even she can only see the sub-packets). Providers in collusion can observe multiple messages from a single innocent provider. If number of such messages is $2c-1$, colluding providers can see all sub-packets of a secret value and thus be able to reconstruct the sub-secret value. If the reconstructed secret value equals 1, providers claim the innocent provider has the term, otherwise, the innocent provider do not have the term. 2) For privacy $P2$, we assume searchers/index server search for term t and claim any provider in the result list has documents of term t.

4.2 Privacy Characteristic

We analyze privacy characteristic of our protocol against different roles. Our protocol achieves privacy preserving in many situations, like against network eavesdroppers and single semi-honest provider. There are also certain scenarios in which privacy could be possibly leaked, as the two cases stated in our attack model. In this section, we analyze privacy preserving of $P1$ against eavesdroppers, single provider and colluding providers. For privacy $P2$, we will conduct privacy evaluation by experiments.

4.2.1 Network Eavesdropper

For eavesdroppers, our protocol achieves privacy preserving mainly by encrypted communication; All communications in our protocol are authenticated and encrypted. Eavesdroppers seeing a series of cyphertext can not have any knowledge about its content and thus can't make any informed claims. Secure channels[13] guarantees eavesdroppers themselves can not obtain the cryptographic keys.

4.2.2 Single Semi-honest Provider

In the presence of semi-honest providers, our protocol achieves information-theoretic security.[8] A single provider p_i can only see at most one sub-packet/share out of totally $2c-1$ sub-packets of one sensitive sub-secret from other provider $p_{i'}$. The fact that $f'(\cdot)$ is a $(2c-1, 2c-1)$ secret sharing scheme, as proved by Theorem4.1, yields the information-theoretic security. Thus, adversary obtaining one piece/packet learn no information on the value of v_i at all and can not even make informed claim.

[8]Unlike cryptographic security as in secure channel case, information-theoretic security does not rely on any assumptions of computation theory and is less computation-intensive.

THEOREM 4.1. *If $f(\cdot)$ is a (c, c) secret sharing scheme with packets on domain $\mathbb{Z}_q$, $f'(\cdot)$ is a $(2c-1, 2c-1)$ secret sharing scheme on $\mathbb{Z}_q$, too. Specifically,*

- ***Recoverability**: Given $2c-1$ packets (j, y_j)'s where*

$$y_j = \begin{cases} u_{i+j,-j} & \text{if } j \in \{-c+1, \ldots, -1\}, \\ u_i & \text{if } j = 0, \\ u_{i,j} & \text{if } j \in \{1, \ldots, c-1\}. \end{cases}$$

  the secret v_i can be easily reconstructed.

- ***Secrecy**: Given any $2c-2$ or fewer packets, one can learn nothing about value of v_i, in the sense that the conditional distribution given the known packets is the same to the prior distribution,*

$$\forall x \in \mathbb{Z}_q, prob(v_i = x) = prob(v_i = x | \forall j \in I, (j, y_j))$$

  where I is any set with $2c-2$ or less elements in $\{-c+1, -c+2, \ldots, 0, \ldots, c-1\}$.

PROOF. The first condition is directly implied by Equation 4. For the second condition, we take the worst case in consideration, that is, when there are $2c-2$ packets available to adversary. Let $(j', y_{j'})$ denote the only missing packet. We consider two cases: Case 1), $j' \geq 0$. In this case, we can use Equation 3 to reconstruct the value $u_{i,0}$, that is, $u_{i,0} = (y_0 - y_{-c+1} - \cdots - y_{-1}) \mod q$. Then for (c, c) secret sharing scheme f, we have all c packets determined, except for $u_{i,j'}$. By the definition of (c, c) secret sharing, the value $v_i = f(\cdot)$ is completely undetermined. Case 2), $j' \leq 0$. We can determine all $c-1$ packets $u_{i,j}$ for $j \in \{1, 2, \ldots, c-1\}$, and we turn to prove that $u_{i,0} \equiv (y_0 - y_{-c+1} - \cdots - y_{-1}) \mod q$ is completely undetermined given one packet $(j', y_{j'})$ is missing. We further consider two cases, if $j' = 0$, we have $u_{i,0} - y_{j'} \equiv (-y_{-c+1} - \cdots - y_{-1}) \mod q$; otherwise, we have $u_{i,0} + y_{j'} \equiv (y_0 - y_{-c+1} - \cdots - y_{j'-1} - y_{j'+1} - \cdots - y_{-1}) \mod q$. In both cases, the RHS of the equation is completely determined, while LHS, in the form $u_{i,0} \pm y_{j'}$, is not. Applying lemma 4.2, we can see that the distribution of $u_{i,0}$ is fully unaffected by the given knowledge of $u_{i,0} \pm y_{j'}$ (but not $y_{j'}$). Thus, $u_{i,0}$ is completely undetermined. Note $v_i = f(u_{i,0}, u_{i,1}, \ldots, u_{i,c-1})$ is a (c, c) secret sharing scheme, thus the secret v_i is completely undetermined. □

LEMMA 4.2. *Random variable a, b are natural numbers in domain $\mathbb{Z}_q$. Their values are independently chosen and uniformly distributed in $\mathbb{Z}_q$. Then $\forall x, y \in \mathbb{Z}_q$, we have*

$$prob(a = x) = prob(a = x | (a \pm b) \mod q = y) = \frac{1}{q} \quad (7)$$

Proof of Lemma 4.2 can be found in Appendix.

4.2.3 Semi-honest Providers in Collusion

Our protocol is resistent to providers in collusion. In specific, 1) when collusion is of no more than $2c-3$ providers, attackers can't gain any information on v_i $\forall i \in \{1, \ldots, n\}$, as in the single provider case. Hence, our protocol retains the information-theoretic security. 2) When there are more than $2c-3$ colluding providers, privacy could be possibly leaked. In this section, we analyze the possibility and argue it's very unlikely for such breach to occur.

Note sub-packets of a single sub-secret v_i are distributed to at least $2c-2$ providers, which are the $2c-2$ consecutive neighbors of provider p_i. When these nearest-$(2c-2)$ providers happen to form a collusion, then privacy regarding sub-secret v_i is definitely leaked. [9] However, we argue the likehood for collusion of this kind to occur is fairly low. Besides, even in this case, only one sub-secret is disclosed, while other sub-secrets' privacy is still guaranteed. Note in our protocol group members are randomly ordered in ring-like overlay and they follow this protocol specification (since they are semi-honest). The probability for a honest provider to be surrounded by $2c-2$ adversaries is,

$$h = \frac{P_{2c-2}^{m}}{P_{2c-2}^{n-1}} = \frac{m!(n-2c+1)!}{(n-1)!(m-2c+2)!} \quad (8)$$

m is the total number of colluding providers in the network. As can be seen, the probability is very low for moderate value of c. For example, when $m = \frac{n}{2}$, the probability is $h < (\frac{1}{2})^{2c-2}$, which quickly approaches 0 for large n.

In practice, the number of colluding adversaries is usually small. For example, as discovered in a peer-to-peer measurement study [14], most collusions are of two or three mutually colluding nodes. Thus, by setting c at relatively small value (e.g., 5), it suffices to make our SS-PPI secure against colluding attacks.

4.2.4 Index Server aware of Term Selectivity

Our protocol discloses group-wise term selectivity in the published index, rendering privacy $P2$ vulnerable. With this information, index server can make informed decision on picking up the right group and term (if any) and perform security attacks successfully. To overcome this vulnerability in our system, we propose an enhanced version of SS-PPI which preserves privacy $P2$ and still achieves efficiency in performance. In specific, we add noises to the term-wise bit before group aggregation starts. For term t and a possession bit v, the provider generate another number v' in $\{0, 1, \ldots, b\}$ to do aggregation, as follows,

$$v' = \begin{cases} rand(b) \text{ in } \{1, 2, \ldots, b\} & \text{if v=1} \\ 0 & \text{if } v = 0. \end{cases}$$

Now the sum of v' does not necessarily equal the number of positive providers in each group. By this means, privacy can be further preserved, at expenses of extra inaccuracy of ranking between groups in the public index. Parameter b controls the trade-off between meta-data privacy and ranking accuracy. With this approach, the selectivity observed to be high could end up being small ones. In this sense, we prevent an adversary from picking up the term with high selectivity.

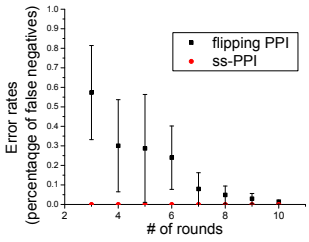
5. EXPERIMENTAL EVALUATION

In this section, we evaluate our SS-PPI, mainly by simulations. The evaluation is based on the comparison to previous work, flipping PPI[2]. Throughout the experiments, we mainly use synthetic dataset which consists of 10^5 providers, which are further mapped to 1000 groups. Groups are disjoint and are configured with an expected group size, that is, $10^5/10^3 = 100$. We also used a peer-to-peer dataset[16], which is developed based on the TREC WT10g web test collection, a 10 gigabyte, 1.69 million document subset of the VLC2 collection[11]. In our default setting, we run each experiments 20 times and report the averaged results.

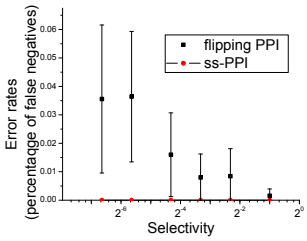
5.1 Correctness

In the first set of experiments, we evaluate the PPI's correctness. For SS-PPI, the correctness is measured by the probability that the aggregated result equals number of positive providers in a group, that is, $\sum_0^{n-1} v_i$; for flipping PPI, it's measured by that for logical OR of v_i's. In the first experiment, we vary the number of rounds and fix term selectivity being 0.1, the results are shown in 4a; then, we test the protocol with terms of different selectivity and fixing the rounds to be 10, results shown in 4b. In general, SS-PPI achieves 100% accuracy, while flipping PPI doesn't. For small number of rounds and selective terms, flipping PPI incurs relatively high inaccuracy and uncertainty. It becomes more accurate as the number of rounds goes up. However, this improvement comes at the expenses of more severe privacy leakage, more bandwidth consumption and longer time duration, as will be shown.

[9] Here, we exclude the trivial case that p_i is itself in collusion, because then v_i can be known locally from p_i.

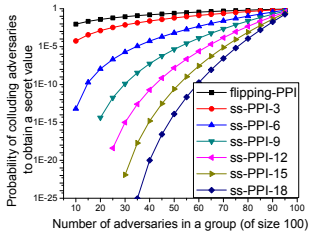


(a) Varying number of rounds

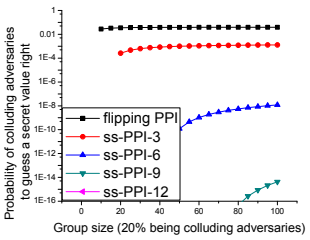


(b) Varying term selectivity

Figure 4: Search correctness



(a) Varying number of adversaries



(b) Varying group size

Figure 5: Privacy preserving against collaboration of adversaries

5.2 Privacy Preserving

This set of experiments evaluate the level of privacy preservation of our SS-PPI. We mainly follow our security analysis to conduct experiments, in which two types of privacy are considered, including index construction privacy $P1$ and published privacy $P2$.

5.2.1 *Preserving Index Construction Privacy $P1$ against Colluding Attacks*

Our simulation for evaluation of privacy preserving against collaborating adversaries is based on Equation 8. flipping PPI can be modeled with $c = 1$ and in our SS-PPI, c ranges from 3 to 18. We have done two experiments; the one is with varying the number of colluding adversaries and fixed group size of 100, the other with varying group size, and the adversaries accounting for 20% of the group. We evaluate the probability of specific positioning of adversaries that leaks privacy. The results are plotted in Figure 5. In general, flipping PPI incurs the highest privacy breach; its leaking probability is an order of magnitude higher than that of SS-PPI, even with small c. As c grows up, the improvement of SS-PPI's privacy preserving against flipping PPI becomes significant. In Figure 5b, the privacy breach generally become more severe, as there are more adversaries. When all providers in a group are malicious, the probability of privacy breach becomes 100%, for all protocols. When there are limited adversaries (which is more likely the case in real world), SS-PPI achieves much better privacy preserving comparing to flipping PPI. As can be observed, there exist a threshold on number of adversaries under which SS-PPI's leaking probability is 0. For example, in the plot, the curve for SS-PPI-18 shows up only when adversaries are more than 35. This is more obvious in Figure 5a. By contrast, flipping PPI is vulnerable to collaborating attacks in all experiment settings. In Figure 5a, we can also see SS-PPI slightly increase privacy leaking probability until certain converging value as group size goes up, while flipping PPI stays constant. Note the probability (y axis) is plotted in log scale, differences between two protocols are significant.

In previous experiment, we consider privacy leakage of one innocent provider. Here, we move forward to study the multiple-provider case. Given a number of colluding adversaries, we measure the number of innocent providers being attacked. Let $a(l)$ denote the minimum number of colluding adversaries required to hack l providers' private information. By analysis model in Section 4, SS-PPI has,

$$a_{SS-PPI}(l) = (c-1)(l+1) \tag{9}$$

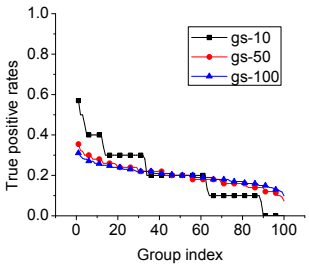
For l and c large enough, the above equation must meet $(c-1)(l+1)+l \leq n$, or $l \leq \frac{n+1}{c} - 1$. For flipping PPI, $c = 2$ and we have,

$$a_{flipping}(l) = l + 1 \tag{10}$$

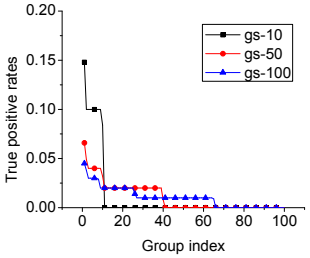
Comparing to flipping PPI, our SS-PPI has higher requirement on the number of colluding adversaries, thus less likely to leak privacy.

5.2.2 *Preserving Privacy $P2$ against Searchers*

In this experiment, we evaluate property of privacy preserving of random grouping. We considered two cases, the common term case and the rare term case. For each case, we measure the true positive rates (i.e., the group-wise selectivity or how many providers in a group do possess the term) under different group sizes. The results are shown in Fig. 6. To visualize the degree of preserving privacy $P2$, we ranked groups based on true positive rates, decently. The maximal level of true positive ratios is critical to the overall degree of privacy preserving, thus being of interests to us. As can be seen from the results, the smaller the group size is, the more non-uniform the distribution of true positive rates is. For instance, the maximal true positive rate for group size of 10 is 0.6 for common terms of selectivity 0.2, while for group size configured to be 100, the maximal true positive rate is around 0.3. In this sense, larger group size leads to more privacy preserving. Comparing rare terms and common terms, grouping with common terms could end up with every group having non-zero true positive rate, implying query broadcast, which is not the case for those rare terms.



(a) Common terms



(b) Rare terms

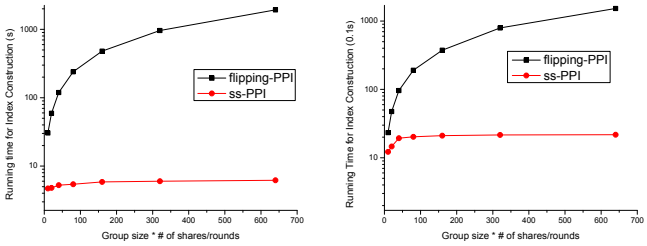
Figure 6: Preserving privacy $P2$ of random grouping

5.3 Performance of Group Aggregation

We evaluate the performance of SS-PPI by simulation. Specifically, we use two distributions on round-trip times (RTTs) to emulate the message delays, that is, gaussian distribution with deviation being 1 and mean being 3, and the distribution modeled from real network traces [12]. In the latter case, about 35% of the messages have RTT $< 50ms$, 60% with RTT $< 100ms$, 25% with RTT $> 200ms$, and the rest are in seconds. For fair comparison, we set $r = c$ (by which the bandwidth costs of SS-PPI are equal to those of flipping PPI). The simulation results are shown in Fig. 7, from which we can see that SS-PPI achieves much better performance in terms of scalability (especially in the large groups). We explain the results based on time-complexity analysis. For SS-PPI, the bandwidth costs are $n \cdot c$ and latency is $max(hop_{i,j})$ for $\forall i \in [0, n-1], j \in [i+1, i+c-1]$. For flipping PPI, the latency is $r \cdot \sum_{i=0}^{n-1} hop_{i,i+1}$ and bandwidth costs are $n \cdot r$ (in terms of number of messages transmitted). When $c = r$, latency of flipping PPI is O($n \cdot r$) while that of SS-PPI is $O(1)$.

5.4 Query Processing Costs

The query processing costs are measured by the number of providers



(a) On network with RTT of gaussian distribution

(b) On network with RTT modeled from real trace data [12]

Figure 7: Time-efficiency of SS-PPI

returned from the index server for a single query. In this experiment, we focus on single-term queries, in which term is randomly picked from index dictionary. We vary the group size from 2 to 1000 and plot the results in Figure 8. Overall, the query costs grow up as the group size increases. For group size bigger than 50, the percentage of providers that need to check by queriers quickly approaches to 100%, implying our index deteriorates to query broadcasting in this setting, which conforms to our previous experimental and analysis result.

We further study the relationship between search recall and search costs. Experiments are conducted in the both cases of common terms and rare terms. Common term is with selectivity 0.1 while rare term with selectivity 0.004, as they can be picked from the peer-to-peer text dataset[16]. Different value of group sizes are picked for experiments. With results illustrated in Fig. 9, we can see that search costs approximately grow in linear to search recall. For 100% recall, search for common terms always require query broadcasting, while search for rare terms only need multicast to the partial set of providers. In rare term case, the search costs are sensitive to the group size; a big group size generally results in more search costs.

The set of experiments give us implication to properly set the value of group sizes. Finding an appropriate value for group size is tricky, because as aforementioned, too big a group size could lead to query broadcast which hurts scalability and performance, while too small a group size deteriorates the level of privacy preserving. From the experiment results, rules of thumb are to set group size to 50, by which only terms with selectivity bigger than 0.01 will ends up with query broadcast (from Fig. 8) and maximal true positive rate for common terms (e.g., with selectivity 0.2) is less than 0.4 (note in this case, the false positive rate is smaller than 0.5, thus meeting the privacy requirement of "Probable Innocence"[2, 18]).

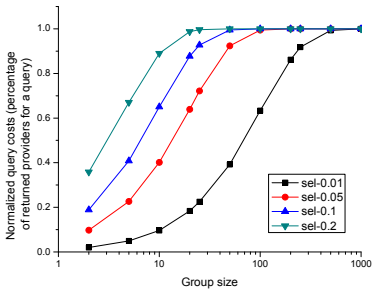
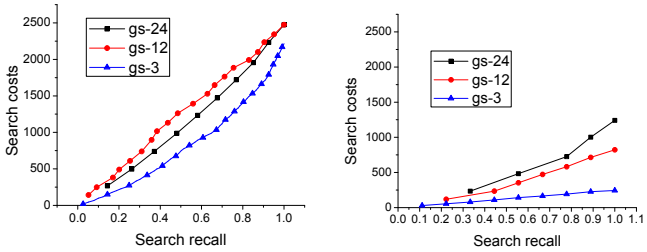


Figure 8: Costs of query answering

6. RELATED WORK

This section briefly surveys relevant work. We review different architectures of privacy-preserving indices proposed in literature, compare two important primitives to construct such architectures, namely secure multi-party computation (SMC) and secret sharing,



(a) Common terms

(b) Rare terms

Figure 9: Search costs in number of providers contacted

and finally discuss other privacy issues in information sharing infrastructures.

6.1 Indexing on Untrusted Servers

The work [2] paved the way for public privacy preserving index from multiple content providers. It randomly organizes providers into disjoint privacy groups; it employs an iterative, randomized algorithm to form group-wise indexing structure, which is further used to direct queries. Comparing to our protocol, this solution suffers three major drawbacks. First, the random grouping strategy makes privacy groups tend to have all terms, and query processing ends up broadcasting. Second, the probabilistic index construction scheme take arbitrary rounds before convergency, leading to unacceptable running time. Third, index construction leaks considerable privacy and is vulnerable to colluding attacks.

Zerber [23] is based on multiple servers with certain amounts of trusts. Zerber distributes inverted index over n servers by using a k-out-of-n secret sharing scheme. During query time, a searcher has to be authenticated and authorized by at least k servers. Then after issuing masked queries individually, searcher obtains shares of matched posting entries from each server, and proceeds to composite the secrets, including indexed terms, document id and relevances. The scheme performs client-side intersection of positing lists, which seriously slow down query performance for large scale dataset (in terms of large number of documents), multiple-keyword queries and searcher with broader accesses. Overall, the amount of trusts assumed on index servers may become unacceptable, since it relies on servers to do authentication and authorization. In particular, access policy privacy is seriously leaked in the sense that any single malicious server cloud disclose such privacy.

Union query dissemination trees (or UQDT) [6] is a distributed privacy preserving index; the indexes are organized as multiple trees, which share the same set of leaf nodes, each corresponding to a disjoint privacy group of publishers/providers. In essence, each QDT can be viewed as a hierarchy of group-wise index at different granularity, and privacy preserving (defined in publisher k-anonymity) is attained in a similar group-wise way. To form the finest group at leaf level, a generic secure multi-party computation [9] is adopted, which however is inefficient and unscalable to thousands of providers. Different QDT's are responsible for different (disjoint) subset of index terms; The multiple-QDT architecture is intended for better load balance and higher throughput, as compared to single UQDT with global set of indexed terms. Specifically, the same physical nodes are intelligently positioned at different levels of different QDT, so overall load is balanced.

6.2 SMC and Secret Sharing

Secure multiparty computation (or SMC) [10] refers to the problem in which multiple parties, each holding a private input, collectively perform a computation without disclosing information more than the output reveals. Many operations in privacy-aware applications [7] can be deemed as secure multiparty computation, including the group-wise index aggregation in our problem. Secret sharing is one primitive for SMC problems. In particular, a generic secret sharing scheme splits a secret into multiple shares, only more

than a certain mount of which can reconstruct the secret. Many secret sharing schemes [19] are additive homomorphic [3]. Our protocol takes advantages of this nice property and applies in privacy preserving index construction. Comparing to other primitives for secure multiparty computation, secret sharing is advantageous; it attains information-theoretic security, robust against colluding attacks. More importantly, secret sharing has been shown to significantly outperform those generic SMC protocols in execution efficiency [5, 4]. Secret sharing has been applied in various contexts, for example, database query processing [8], information aggregation [5] and keyword searches[23]. The above approaches put the computation of secret shares onto multiple third parties while index construction in our SS-PPI involves no third party, which sees better scalability.[10] Other work [21] uses similar secret-sharing protocol to preserve privacy in data mining. However, their way to distribute shares incurs load imbalance and hurts performance.

7. CONCLUSION

In this paper, we propose SS-PPI, an efficient and strong privacy preserving index over multiple content sources. Specifically, SS-PPI adopts a new architecture of PPI, namely role-sensitive PPI, that takes into account role access information and achieves better search performance. We also identify a new type of potential privacy leakage in this architecture, that is, access policy privacy. Further, we propose a secret sharing based approach for efficient and secure construction of public index, from multiple content providers. Comparing to previous work, SS-PPI makes a better balance between privacy preserving and search performance. Our protocol is secure against colluding adversaries and achieves information-theoretic privacy preserving. We also conduct extensive analysis and experiments that show advantages of SS-PPI in terms of query performance and security properties.

Acknowledgements

The authors would like to thank the anonymous reviewers. This work is partially supported by grants from NSF CISE NetSE, NSF CyberTrust, an IBM SUR grant, an IBM faculty award, and a grant from Intel Research Council.

8. REFERENCES

[1] Nhin: http://www.hhs.gov/healthit/healthnetwork.
[2] M. Bawa, R. J. Bayardo, Jr, R. Agrawal, and J. Vaidya. Privacy-preserving indexing of documents on the network. *The VLDB Journal*, 18(4), 2009.
[3] J. C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret sharing. In *CRYPTO*, pages 251–260, 1986.
[4] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, pages 192–206, 2008.
[5] M. Burkhart, M. Strasser, and X. A. Dimitropoulos. Sepia: Security through private information aggregation. In *USENIX Security*, 2010.
[6] E. Curtmola, A. Deutsch, K. K. Ramakrishnan, and D. Srivastava. Load-balanced query dissemination in privacy-aware online communities. In *SIGMOD*, 2010.
[7] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *NSPW*, pages 13–22, 2001.
[8] F. Emekçi, D. Agrawal, A. E. Abbadi, and A. Gulbeden. Privacy preserving query processing using third parties. In *ICDE*, page 27, 2006.
[9] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, 1987.
[10] S. Goldwasser. Multi-party computations: Past and present. In *PODC*, pages 1–6, 1997.
[11] D. Hawking. Overview of the trec-9 web track. In *TREC*, 2000.
[12] H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times. *Computer Communication Review*, 32(3):75–88, 2002.
[13] B. W. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Trans. Comput. Syst.*, 10(4):265–310, 1992.
[14] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the maze p2p file-sharing system. In *ICDCS*, 2007.
[15] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.
[16] J. Lu and J. P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, pages 199–206, 2003.
[17] H. Pang, X. Ding, and X. Xiao. Embellishing text search queries to protect user privacy. In *VLDB*, 2010.
[18] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.
[19] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
[20] M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *CCS*, 2005.
[21] S. Urabe, J. Wang, E. Kodama, and T. Takata. A high collusion-resistant approach to distributed privacy-preserving data mining. In *Parallel and Distributed Computing and Networks*, pages 307–312, 2007.
[22] L. Xiong, S. Chitti, and L. Liu. Topk queries across multiple private databases. In *ICDCS*, pages 145–154, 2005.
[23] S. Zerr, E. Demidova, D. Olmedilla, W. Nejdl, M. Winslett, and S. Mitra. Zerber: r-confidential indexing for distributed documents. In *EDBT*, pages 287–298, 2008.

[10] It implies the more providers participate in the network, the more secure we will end up with, because adversaries now need to compromise all providers to obtain all secrets.

APPENDIX

A. PROOF OF THEOREM

THEOREM A.1. *Suppose packets are defined on $\mathbb{Z}_{q'}$ and secrets on $\mathbb{Z}_q$. If $q' = k \cdot q$ (where k is an integer) and the first $c-1$ packets are chosen randomly and independently, $f(\cdot)$ defined by Equation 2 is a (c, c) secret sharing scheme. Formally,*

***Recoverability**: Given c sub-packets $u_{i,j}$ with $j \in \mathbb{Z}_q$, the secret v_i can be easily reconstructed.*

***Secrecy**: Given $c-1$ or fewer sub-packets, the secret v_i is completely undetermined, in the sense that each possible value on sub-secret domain is equally likely for v_i,*

$$\forall x \in \mathbb{Z}_{q'}, \qquad prob(v_i = x) = prob(v_i = x | \forall j \in I, u_{i,j} = y_j) = \frac{1}{q'}$$

where I is any set of $2c-2$ or less elements in $\{-c+1, \ldots, 0, \ldots, c-1\}$.

PROOF. We consider the worst case, that is, when $2c-2$ $u_{i,j}$ are known in priori and there is only one packet variable that is unknown. Denote the unknown variable and the sum of the known variables by u_x and v_i', respectively. Then, $v_i = v_i' + u_x \mod q$. For any value $v_i = x$ in $\mathbb{Z}_q$ and fixed v_i', there is exactly k values in $\mathbb{Z}_{q'}$, subject to $v_i = v_i' + u_x \mod q$. And it's easy to see for different x, the set of these k values are disjoint (essentially, forming an equivalence class). The probability for $v_i = x$ is then the probability for u_x to fall in the equivalence classes corresponding to x. Since u_x is unknown and uniformly distributed, probability for $v_i = x$ stays the same, even when $2c-2$ $u_{i,j}$ is known in advance. The theorem holds.

B. PROOF OF LEMMA

Lemma 6.2 Random variable a, b are natural numbers in domain $\mathbb{Z}_q$. Their values are independently chosen and uniformly distributed in $\mathbb{Z}_q$. Then $\forall x, y \in \mathbb{Z}_q$,

$$prob(a = x) = prob(a = x | (a \pm b) \mod q = y) = \frac{1}{q} \qquad (11)$$

PROOF.

$$\begin{aligned} & prob((a \pm b) \mod q = y | a = x) \\ &= prob(b = (y \mp x) \mod q | a = x) \\ &= prob(b = (y \mp x) \mod q) = \frac{1}{q} \end{aligned}$$

The derivation is due to the fact that variable a and b are mutually independent.

$$\begin{aligned} & prob(a = x | (a \pm b) \mod q = y) \\ &= \frac{prob(a = x \wedge (a \pm b) \mod q = y)}{prob((a \pm b) \mod q = y)} \\ &= \frac{prob(a = x) \cdot prob((a \pm b) \mod q = y | a = x)}{prob((a \pm b) \mod q = y)} \\ &= \frac{prob(a = x) \cdot prob(b = (y \mp x) \mod q | a = x)}{\sum_{x'=1}^{q} prob(b = (y \mp x') \mod q | a = x') \cdot prob(a = x')} \end{aligned}$$

Because a,b are independent random variables, so $\forall x'$, $prob(b = (y \mp x') \mod q | a = x') = prob(b = (y \mp x') \mod q) = \frac{1}{q}$. And since $prob(a = x') = \frac{1}{q}$, we have

$$\begin{aligned} & prob(a = x | (a \pm b) \mod q = y) \\ &= \frac{\frac{1}{q} \cdot \frac{1}{q}}{\sum_{x'=1}^{q} \frac{1}{q} \cdot \frac{1}{q}} = \frac{1}{q} \\ &= prob(a = x) \end{aligned}$$