EAI Endorsed Transactions
on Security and Safety

Research Article **EAI.EU**

Privacy-Preserving Multi-Party Directory Services

Yuzhe Tang[1,*], Kai Li[1], Katchaguy Areekijseree[1], Shuigeng Zhou[2], Liting Hu[3]

[1]Department of EECS, Syracuse University, NY, USA
[2]School of Computer Science, Fudan University, China
[3]School of Computing and Information Sciences, Florida International University, FL, USA

Abstract

In the era of big data, the data-processing pipeline becomes increasingly distributed among multiple sites. To connect data consumers with remote producers, a public directory service is essential. This is evidenced by adoption in emerging applications such as electronic healthcare.

This work systematically studies the privacy-preserving and security hardening of a public directory service. First, we address the privacy preservation of serving a directory over the Internet. With Internet eavesdroppers performing attacks with background knowledge, the directory service has to be privacy preserving, for the compliance with data-protection laws (e.g., HiPAA). We propose techniques to adaptively inject noises to the public directory in such a way that is aware of application-level data schema, effectively preserving privacy and achieving high search recall.

Second, we tackle the problem of securely constructing the directory among distrusting data producers. For provable security, we model the directory construction problem by secure multi-party computations (MPC). For efficiency, we propose a pre-computation framework that minimizes the private computation and conducts aggressive pre-computation on public data. In addition, we tackle the systems-level efficiency by exploiting data-level parallelism on general-purpose graphics processing units (GPGPU). We apply the proposed scheme to real health-care scenarios for constructing patient-locator services in emerging Health Information Exchange (or HIE) networks.

For privacy evaluation, we conduct extensive analysis of our noise-injecting techniques against various background-knowledge attacks. We conduct experiments on real-world datasets and demonstrate the low attack success rate for the protection effectiveness. For performance evaluation, we implement our MPC optimization techniques on open-source MPC software. Through experiments on local and geo-distributed settings, our performance results show that the proposed pre-computation achieves a speedup of more than an order of magnitude without security loss.

Received on 15 December 2018; accepted on 20 January 2019; published on 29 January 2019

Keywords: Secure Multi-party Computation, Public Directory, Background-knowledge Attacks

doi:10.4108/eai.29-7-2019.159627

1. Introduction

In the era of big-data, personal data is produced, collected and consumed in digital forms, bringing unprecedented convenience to the society. As data production and consumption are distributed at different sites, sharing person-specific data over the Internet becomes a popular application paradigm as widely observed in a variety of domains ranging from electronic healthcare, social networks, Internet of things, malware detection, to many others.

A public directory service is a crucial data-sharing component. In a data-sharing pipeline, a data consumer queries the directory service to locate the producer sites that may have the documents of interest. The directory service maintains the private

*Corresponding author. Email: ytang100@syr.edu producer-location information, and connects data consumers and producers. For instance, in electronic healthcare, HIE or Healthcare Information Exchange is an emerging data-sharing platform [4, 9] where the directory called locator service [1, 5, 8, 11] helps a doctor (data consumer) find the electronic medical records (EMR) of a visiting patient (data producer). The data-location information ("which hospitals a patient has visited") may reveal privacy-sensitive facts; for instance, knowing that a celebrity visited a rehabilitation center, one can infer that s/he may have a drug problem.

A naive way of constructing the directory is for any data producer to directly publish its list of associated people (e.g., the list of patients having visited a hospital). However, this approach discloses the private data-location information to network adversaries performing traffic analysis. This privacy disclosure leaks "identifiable information" and would violate data-protection laws (e.g., HIPAA in USA [6], EC95-46 in European Union [3] and various privacy laws in Asian countries [49]) that govern the data-sharing across borders in regulatory domains.

This work aims at providing a practical public directory service over the Internet. We address two salient features: 1) privacy preservation against background-knowledge attacks, and 2) secure and efficient multi-party construction of the directory among distrusting data producers. In the following, we elaborate on the two features and formulate the research.

First, with the directory data bearing privacy and being exposed to the public, it is imperative to preserve privacy. Existing data-privacy and anonymization techniques are inadequate for preserving privacy in the public directory. Differential privacy [31] is for the use in a statistics database but is insufficient for protecting directory data where the published data retains the individual records (not some statistical digests). Data anonymization notions such as k-anonymity [59], l-diversity [59], t-closeness [45] preserve privacy at a finer granularity, but is not specific to the background knowledge attacks in domain applications. In addition, the random noises used in existing techniques are insufficient as it lacks the needed models to make the noises indistinguishable (from true positives) in the presence of background knowledge.

We propose to model the background-knowledge exploited in the domain applications and accordingly select noises such that they are indistinguishable with true positive records. We propose a top-k similarity algorithm for selecting the indistinguishable noises in background knowledge attacks. Specifically, we aim at making the distribution of noises similar to that of true positives. The similarity is measured on the dimension of external, public knowledge. For instance, in HIE, the similarity between hospitals (producers) can be defined by hospital specialties and geographic locations. The top-k computation strikes a balance between the practical computation complexity and the needs to counter the background knowledge attacks of exponentially growing possibilities. While our background-knowledge modeling may not be exhaustive, our noise-selecting approach is effective in HIE and other relevant scenarios as evaluated in our study based on real-world data.

Second, we address the multi-party construction of the directory service from data producers. A defining characteristic is that data producers operate autonomously and generally do not trust each other (e.g., hospitals who compete on the same customer basis). Yet, producers need to publish their private data to construct the directory service. To guarantee the provable security, one can model the directory-construction problem as a secure Multi-Party Computation (MPC) problem [24, 26, 34, 46, 64] where a joint computation with inputs private to different parties is evaluated securely. A naive instantiation of the directory publication is by embedding entire publication logic in an MPC protocol, which however causes high overhead and is impractical, because of the expensive cryptographic primitives used in constructing an MPC. A conventional remedy is to identify the private part of the computation (e.g., by data-flow analysis [17, 52]) and to map only this part to the MPC. Unfortunately, this approach is not effective in our problem, as the private and public data flows of the directory-construction logic are inter-tangled and separating them becomes difficult.

We tackle the efficiency of secure multi-party directory construction. We do so at both the protocol level by proposing pre-computation techniques and the systems level by exploiting data-level parallelism. Concretely, we propose an aggressive pre-computation technique that minimizes (instead of separating) the private computation for multi-party directory publication. Concretely, we conduct the pre-computation by considering all possible values of private data. It then applies expensive MPCs to a simple selection logic, that is, select from the list of pre-computed results by the actual value of private data. At the first glimpse, this optimization technique may seem counter-intuitive as the pre-computation augments the input space exponentially. In practice, particular to our directory construction problem, its effectiveness relies on the application characteristic: The public computation is usually bulky and private identity data is much smaller. For instance, achieving the privacy of t-closeness [45] entails complex computation on the public background knowledge, such as similarity/distance calculation. In addition, we propose several policies that vary in the degree of pre-computation aggressiveness. The policies can help the optimization technique adapt to concrete scenarios with different private-data sizes.

To improve the system-level efficiency, we exploit the data-level parallelism native in multi-party construction and implement the pre-computation on General-Purpose Graphics Processing Units (GPGPU). We implement our design on real MPC software [24] and conduct performance evaluation in both local and geo-distributed settings. Our evaluation verifies the pre-computation speedup by more than an order of magnitude over the conventional approaches. Through evaluation on real-world datasets, the assurance of privacy preservation is also verified.

The contributions of this work are listed as following:

- We address the privacy preserving of directory data under background knowledge attacks. We model the background knowledge and propose techniques to generate indistinguishable noises even with background knowledge.
- We analyze our noising techniques with various background-knowledge attacks. We also conduct an empirical evaluation on real-world datasets and demonstrate the effectiveness of the protection.
- We address the security and efficiency in multi-party directory construction. We propose secure multi-party pre-computation and tailor it for the directory construction. The observation is that the public background knowledge in directory publication can be isolated from expensive multi-party computation (MPC). We implement this optimization design on real MPC software.
- We propose a systems-level optimization technique for efficient directory construction. The optimization is by conducting data-parallel pre-computation and by implementing it on GPGPU.
- We conduct performance evaluation and demonstrate an order of magnitude performance speedup.

The rest of the paper is organized as following: § 2 formulates the research problem. It presents background-knowledge attacks in § 3 and noising-based protection techniques in § 4. The directory construction with efficiency is presented in § 5. Performance evaluation is presented next in § 6. § 7 surveys the related work and § 8 concludes the paper.

2. Research Formulation

This section presents the system and threat model, the security goals, survey of existing techniques, and preliminary on privacy-preserving data publication algorithms.

2.1. System Model

The target eco-system involves three roles: data producers, data consumers, and the host of directory service. Each data producer owns a table of personal records where each record is keyed by the identity of the owner of this record. Given a person of interest, a data consumer would want to find his/her records at all producer sites. The directory service helps the consumer "discover" relevant data producers who maintain the result records.

Formally, sharing personal records in our system works in two steps: First, a data consumer interested in a person's records poses a query to the directory service and looks up the list of producers who have this person's records. Then, the consumer contacts individual producers and locally searches the records there. In this process, the query is based on a personal identity, which we assume is known globally. In practice, this global identity can be maintained physically by an identity-management server or constructed virtually such as by patient record linkage in healthcare [40, 62].

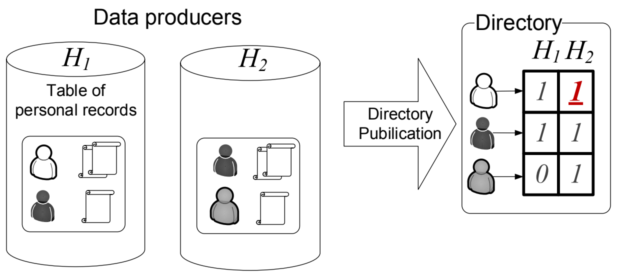


Figure 1. System model of public directory: Two data producers share three people's records. In the directory, value one means presence and zero means absence (e.g., producer H_1 does not have gray person's records). The underscored one in red is a false positive in the sense that producer H_2 does not have the record of the white person but the directory records the opposite (for the sake of privacy preservation).

We assume each data producer locally has a data-protection mechanism in place (e.g., user authentication and authorization) that prevents an external party from accessing the records without the data owner's consent. Figure 1 illustrates the abstract model of our system. The model is applicable to data-sharing applications in regulatory domains; A concrete scenario is about sharing patient electronic medical records (EMR) in healthcare information exchange networks, where data producers are hospitals, personal data are patients' EMRs and consumers can be physicians diagnosing patient. The details of the scenario will be elaborated in § 2.4.

The target computation of this work is about building the directory. A baseline is that each data producer sends its local access-control list to the third-party directory which enforces the access control when serving the directory requests. This baseline however becomes problematic when the directory host is untrustworthy (e.g., by third-party clouds): First, enforcing access control with integrity entails user authentication and authorization to be done by a trusted party. Second, the local access-control list reveals the binding between a person and her data producers, which can be privacy-sensitive in many applications. For instance, in Healthcare scenarios, the binding between a patient and a rehabilitation center can reveal that this person may have a drug problem. Even when the directory is protected by the host, an adversary can easily recover the binding by performing network traffic analysis and extracting this information from the side-channel of the consumer access trace.

The life cycle of the public directory is of two stages: serving consumers' online requests and being constructed from multiple data produvers. In the following, we present the security/privacy goals respectively in the two stages.

2.2. Privacy of Serving the Directory

The directory is served in the public with all containing data exposed. As mentioned, the directory data bears privacy and can be a target attracting adversaries. While there are existing data-privacy definitions, such as k-anonymity [59], l-diversity [59], t-closeness [45], we consider the background-knowledge attacks. The attack leverages external knowledge about the data producers and personal records to distinguish the noises in the privacy notions. In § 3 we present the detailed data model and background knowledge used in attacking the public directory service.

The privacy preservation of public directory under background knowledge requires that the false positive producers are indistinguishable from true positives, such that the distribution of true positives is similar to that of false positives. What's noteworthy is that the similarity is measured on the dimension of external, public knowledge. For instance, in HIE, the similarity between hospitals (producers) can be defined by hospital specialties and geographic locations. In this work, we mainly use the notion of ϵ-privacy [61] to drive further presentation. The main idea of ϵ privacy is to bound the number of noises or false positives in the published list of producers by a percentage of ϵ.

2.3. Security of Multi-Party Directory Construction

When constructing the directory, it involves with multiple mutually distrusting producers. In our problem, a data producer runs autonomously and distrusts external parties including peer producers. Data producers get engaged in the distributed computation for publishing privacy-preserving directory where they exchange information with each other.

In the threat model, an adversary can eavesdrop all messages being exchanged during the distributed directory publication. For a producer, the adversary can be a network eavesdropper or a peer producer. Formally, this is the semi-honest model used in formulating a secure multi-party computation problem [21], where the adversary, being a participant in the computation, honestly follows the protocol execution but is curious about any data that flows through her during the execution. Multiple adversaries may collude. Given a network of n producers, we consider the collusion can be up to $n-1$ peer producers.

The security goal is to assure the data security in the directory-publication process. Our security goal is to ensure perfect privacy (in an information-theoretic sense). Informally, it means an adversary's view only depends on her input and public output. In other words, the messages exchanged in the protocol execution when the input of other parties take one value are "indistinguishable" from those when the input of other parties take another value. More formal treatment of the MPC data security can be found on classic texts [26].

Our threat model and security goal fit in the real-world requirement for policy compliance in data sharing. In many regulatory domains, a data producer has the responsibility of protecting the personal data it maintains and complying data-protection laws. For instance, HIPAA [6] states any identifiable information about a patient cannot be shared to any third-party, without the patient's consent.

Non-goals of this work include directory data authenticity, producer-site data protection, key management, etc. Encrypting data on the directory is orthogonal, as the content of directory is anyway disclosed to the adversary of network eavesdropper performing traffic analysis.

2.4. Applications: Healthcare Locator

One of motivating applications of P^3I is the public locator service in healthcare information exchange networks (HIE). HIE is a health data-sharing network where the data is patient electronic medical records (EMR), data producers are hospitals where each patient visit results in the generation of new entries in an EMR, and data consumers are clinical doctors. A typical application scenario is effective sharing patient's EMR during a clinical visit where the doctor diagnosing a patient needs to view the relevant EMRs of the patient which are produced and stored in remote hospitals.

This scenario features privacy-sensitive data. Patient EMRs are personal, privacy-sensitive documents, the sharing of which must comply HIPAA [6]. Each hospital has its local information-security infrastructure in place (e.g., access control and user authentication).

A directory service, called HIE locator, can be used to facilitate the EMR sharing between hospitals and to help discovery of a patient's previous hospitals. In the normal case, the list of hospitals is discovered by the clinical doctor asking for it. However, this is error prone (e.g., the patient forgets about it) and is inapplicable in emergency (e.g., the patient is sent to hospital unconscious). The privacy-preserving directory can help automate the data discovery and complement the workflow offline to improve the quality of healthcare.

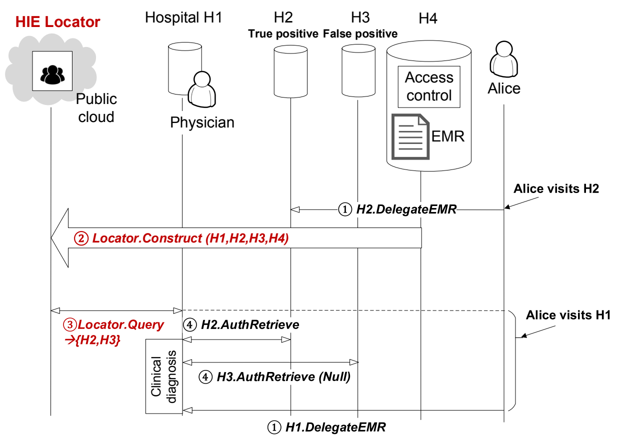


Figure 2. Data-sharing workflow in the HIE: The figure illustrates how Alice's medical record (EMR) stored on Hospital H_2 is used. It shows the entire life cycle of this EMR: The EMR was produced when Alice paid a clinical visit to H_2 (1). During the current visit in Hospital H_1, Alice's physician requires accessing her EMR in H_2 (3). The physician first contacts the third-party locator service hosted on a public cloud (which is constructed at an earlier time (2)) and obtains the list of candidate hospitals H_2 and H_3. Here, H_3 is a noise for privacy preservation purpose. The physician then contacts both H_2 and H_3, and find EMR on H_2 (4). Note that the physician can do so because she has the credential to access data on both H_2 and H_3. For an adversary obtaining the list of H_2 and H_3, she cannot distinguish which hospitals are noise as she does not have the credential.

Figure 2 illustrates the abstract workflow of sharing EMRs in HIE networks. In a clinic scenario, Alice, the patient, is seeing a physician (data consumer) who interacts with HIE network (directory) to locate the hospitals Alice visited before (data producer). In real HIE applications, the locator service runs healthcare software (e.g., OpenEMPI [11]) and is hosted by Amazon AWS alike public clouds. The public clouds are not trustworthy and it entails the use of our privacy-preserving directory protocol for publishing the HIE locator. Concretely, the life cycle of an EMR, including the data-sharing flow, can be divided into three stages: 1) EMR production where Alice's EMRs are generated or updated to reflect her clinical visit; here we assume Alice has given consent on delegating the EMR to the "producer" hospitals. 2) Locator (periodical) publication where the EMR updates are published to the public directory of HIE locator in a privacy-preserving fashion. This is when our directory publication protocol is being invoked in the overall HIE workflow. 3) Locator service where the locator serves the physician's request to locate Alice's producer hospitals (3.1) and find the EMRs of interest there (3.2). In particular for stage 3.2, after the physician obtains the list of potential hospitals (including both true and false positive ones), he will contact each hospital and find EMRs by going through the local user authentication and access control there.

3. Background-Knowledge Attacks

3.1. Attack Framework

Background knowledge: In this work, the background knowledge (B) includes patient demographic information (e.g., age, gender, home address) and hospital profiles (e.g., specialties and location). Specifically, we represent the profile of each hospital by two metrics, a specialty vector and its geographic location (e.g., longitude and longitude). The specialty vector is a vector of ranking scores of the hospital in all specialty categories. The background knowledge we consider in this work is realistic and can be obtained from public data sources; for instance, the hospital profile in terms of location and specialties is public information available on the USNEWS website [7]. And patient demographic information is from various online census datasets [10].

Defense by noising: P^{3}I.Query(p) presents false positive hospitals, serving as noises,[1] to obscure the identities of true positive hospitals which are privacy-sensitive to patient p. The definition of true/false positive and negative hospitals are the following.

Definition: For patient p, a hospital that she visited is defined to be a *truly positive* hospital, denoted by TP. The set of all true positive hospitals is denoted by $I^0 = \{TP\}$.

A hospital that the patient has never visited before is defined to be a *negative* hospital, denoted by N.

Definition: In the P^{3}I, a noise hospital is a hospital which the patient did not visit but the P^{3}I claims that the patient visited. A noise hospital is a false positive and denoted by FP.

[1]Noise and false positive are interchangeable in this paper.

A hospital that appears positive in the P^{3}I can be a true positive or a false positive, and the set of positive hospitals is denoted by $I = \{P\} = \{TP\} \cup \{FP\}$.

Table 1. Notations

P: positive hospital	FP: true-positive hospital
N: negative hospital	TP: false-positive hospital
p: patient	$I^0 = \{TP\}$: true-positive hospitals

ϵ-privacy goal: Given the EMR location of a patient to a list of positive hospitals, one type of information leakage that is inevitable to achieving 100% search recall is that the adversary knows "all the true-positive hospitals are in the P^{3}I.Query result." Beyond that, we assume there is no direct information leaked on a patient's visited hospitals. For instance, the adversary does not know the total number of hospitals visited by a patient. Our privacy-preservation goal is to achieve ϵ-privacy for all considered attacks:

Definition: Given an attack that makes a probabilistic claim, ϵ-privacy is defined to be that the success rate of the attack is statistically upper bounded by ϵ.

Background knowledge attacks: The information flow of an attack is that the adversary can use the publicly available P^{3}I to "reversely" infer the true-positive EMR location I^0, and then from I^0 (or I in some cases) infer the sensitive disease information of the patient. The information flow is $I[\overset{a}{\mapsto} I^0] \overset{b}{\mapsto} disease$ where [] means optional.

The background knowledge can facilitate the attack and can be used in two places in the attack information flow: 1. **Inferring disease in step** b: Knowing hospital specialties can assist in step b to infer the patient disease. The information flow by this attack is $I/I^0, B \overset{b}{\mapsto} disease$, where B represents background knowledge. 2. **Identifying noises in step** a: The background knowledge can be used in step a to distinguish true positive hospitals from the noises. The information flow by this attack is $I, B \overset{a}{\mapsto} I^0 (\overset{b}{\mapsto} disease)$.

3.2. Concrete Attacks

Under the above attack model, we consider three specific attacks. They are classified by attack information flow and background knowledge (as in Table 2). Attack I does not rely on any background knowledge and aims at recovering true positive hospitals in step a. Attack II exploits background knowledge on the hospital specialty and aims at inferring patient diseases in step b. Attack III exploits various background knowledge on hospital and patient profile and aims at recovering true positive hospitals in step a. For different attacks, we present different top-k policies and analyze how the same ϵ-privacy assurance is achieved by P^{3}I.

Table 2. Attacks: The considered attacks are classified by the type of background knowledge used in the attack and the information flow through which the adversary gets to the privacy-disclosing fact on "the patient's disease."

Attacks	Background Knowledge	Target in info. flow
Attack I	$\varnothing$	Step a
Attack II	Specialty	Step b
Attack III	Hospital profile, patient demographic info	Step a

Table 3. Modeling P^{3}I data and background-knowledge: This table describes a scenario that involves one patient and five hospitals that appear to be positive in P^{3}I, $h_1, \ldots, h_5$. We consider two cases: one that all five hospitals are true positives, and one that among the five, h_2 and h_3 are the true positives. Background knowledge about hospital specialty implying patient gender, and geographic distance between hospital and patient home is presented. We also show the non-matching scores (m_e and m_f) on different background knowledge.

Hospitals	h_1	h_2	h_3	h_4	h_5
Case 1	TP	TP	TP	TP	TP
Case 2	FP	TP	TP	FP	FP
Specialty	Cancer	Rehab	Cancer	Woman's center	Rehab
Gender	F/M	F/M	F/M	F	F/M
m_e	1	1	1	0	1
Geo-distance	0.4	0.5	1	4	5
m_f	2.5	2	1	.25	.2

Attack I. In Attack I, the adversary randomly picks one hospital from $\{h_1, \ldots, h_5\}$ without any external knowledge, and makes a claim that the patient visited the hospital. The claim, if it's true, leaks the sensitive information (knowing a patient visits a rehabilitation center discloses her drug addiction problem). As illustrated by case 1 in Table 3, when all five hospitals are true positive, the claim is always true and any type-I attack always succeeds.

Attack II. With the background knowledge of hospital specialties, the adversary can infer the health condition of the patient. The attack follows the information flow: $I, B \overset{b}{\mapsto} disease$.

The attack is successful when all positive hospitals end up with few specialties. Consider the extreme case that all positive hospitals are of the same type, say "rehabilitation centers". Then, no matter which hospital is true positive, the adversary can be certain that the patient must have an addiction-related problem. Note that this is different from attack I where the hospital in the claim must be true positive to have the attack to succeed.

Attack III. In Attack III, the adversary takes on step a to distinguish noise and true-positive hospitals by exploiting the knowledge on patient and hospital profiles. The patient profile in consideration is her demographic information such as home address, gender, age groups, etc. And the hospital profile includes the hospital's specialties, location, etc.

The attack works by the common knowledge on linking patients and hospitals. For instance, given a male patient, the adversary can easily determine that a woman's health center showing up in a P^{3}I search result must be a false positive. Likewise, to a teenage patient, a hospital specialized in geriatrics is unlikely to be true positive. A hospital in the New York State is probably a false positive to a patient living in the State of Georgia. In general, the "non-matching" relationship through background knowledge can assist to reveal the identity of a noise hospital, thus improving the attack success rate. We formulate the relationship by non-matching score m.

Definition: Given the background knowledge B on patient p and negative hospital N, the non-matching score $m_B(p, N)$ measures the unlikelihood that the patient has visited the hospital. The non-matching score can also be expressed between a true positive hospital P and a negative hospital N, as $m_B(P, N)$.

Depending on the application scenarios, the non-matching score can take various values.

- **Exact-match**: The non-matching score takes a binary value, indicating whether the hospital matches the patient. In the previous example involving patient gender and hospital specialty, the non-matching score is 1 (0) when a male (female) patient and a woman's health center are considered. The implication of the non-zero score is that the woman's health center should not be chosen as a noise for a male patient.

- **Fuzzy-match**: The non-matching score takes values that are continuous. In the previous example involving a New York hospital and a patient in Georgia, the non-matching score is measured by the geographic distance between the two. The intuition is that the more distant a hospital is to the patient's location, the less likely there is a match between the two. The implication is that a hospital too far away from a patient should not be chosen as a noise for the patient.

4. Attack Mitigation with Centralized Directory Construction

We now describe P^{3}I.Construct($\{\forall H\}$), that is, how P^{3}I is constructed with the required level of noises. In this section, we consider the centralized P^{3}I construction by assuming a hypothetical authority that exists and is trusted by all hospitals. We will remove this assumption and present the secure and distributed P^{3}I construction in the next section.

Asymmetric Deterministic Response: In the P^{3}I construction, we allow a negative hospital to be published as a false positive. On the other hand, we do not allow false negatives, that is, a true positive hospital will always be published as positive. This rule, illustrated in Formula 1, leads to 100% search recall[2] and prevents any legitimate hospital from escaping the search result.

We call this publication primitive by asymmetric deterministic response (or ADR), reminiscent of the classic randomized response [63]. Comparing the randomized response, our ADR is *asymmetric* in that it treats binary input (N or P) differently or "asymmetrically", and is *deterministic* in that it flips input 0 based on certain deterministic conditions (described by top-k policies in § 4.1).

$$
\begin{aligned}
ADR(N) &\rightarrow \begin{cases} P, \text{chosen as noise by Algo. 1} \\ N, \text{otherwise} \end{cases} \\
ADR(P) &\rightarrow P
\end{aligned}
$$

Top-k Algorithm: Given a patient whose location information to hospitals is I^0 (i.e. the list of visited hospitals), the P^{3}I construction problem boils down to *noise generation*, that is, properly choosing a certain number of false positive hospitals. To a specific patient, the selection favors negative hospitals that may or may not be similar to the set of true positive hospitals (as will be discussed in § 4.1). We thus define a hospital-to-hospital-set distance, $D(N, I^0)$, which measures the dis-similarity between a negative hospital N and the set of all true positive hospitals of a patient, I^0. The selection stops when certain condition is met. The top-k algorithm is illustrated in Algorithm 1.

Listing 1: topk(I^0)
Sensitive input I^0: true positive hospitals visited by a patient
Non-sensitive output $I = \{FP\} \cup I^0$: all positive hospitals

```
$\{FP\} \leftarrow NULL$
$\{P\} \leftarrow I^0$
WHILE(stop-condition($\{FP\}$))
    Find $h_{im}$ s.t.
        $D(h_{im}, \{P\}) = \min_{\forall i \notin I^0 \cup \{FP\}} D(h_i, \{P\})$
    $\{FP\}$.add($h_{im}$)
    $\{P\}$.add($h_{im}$)
```

[2]Search precision in P^{3}I is sacrificed for better privacy preservation. The implication of low search precision is that there are extra hospitals the record-searcher needs to contact.

```
RETURN  $\{FP\} \cup I^0$
```

4.1. Mitigation and Security Analysis

ϵ-Privacy under Attack I. Recall that the top$k$ function in Algorithm 1 exposes two methods to configure: stop condition and distance definition; we call those two top-k policy. To mitigate Attack I, we use a top-k policy as below. Here, we re-use the notation of FP to denote the number of false positive hospitals. The distance is simply set to constant 1 which gives negative hospitals equal chances to be chosen as noise.

Attack-I mitigation:

- **Top-k stop condition**:

$$FP \geq TP(\epsilon^{-1} - 1) \tag{1}$$

- **Distance definition**: $D_I(N, \{P\}) = 1$

Proposition: P^{3}I can mitigate Attack-I with assured ϵ-privacy.

Proof. From the top-k stop condition in Equation 1, we can have the following.

$$\begin{aligned} & FP \geq TP(\epsilon^{-1} - 1) \\ \Rightarrow & \frac{TP}{FP + TP} \leq \epsilon \end{aligned} \tag{2}$$

Given Attack I follows the information flow $I \overset{a}{\mapsto} I^0$, the success rate is:
$Pr(I^0|I) = \frac{Pr(\{TP\})}{Pr(\{TP\}\cup\{FP\})} = \frac{TP}{TP+FP} \leq \epsilon$
ϵ-privacy thus holds. □

ϵ-Privacy under Attack II. **Straw-man by l-Diversity**: Attack II might be mitigated by l-diversity [48] which in the P^{3}I context works by making a patient's diseases "anonymous" among l alternative diseases. However, l-diversity does not automatically lead to ϵ-privacy: While the former has restricts the number of different specialties, the latter restricts the number of negative specialties.[3]

ϵ-Privacy Assurance The intuition of the protection is to choose enough false positive hospitals such that the false positive specialties suffice to bound the rate that the adversary can successfully pick a true-positive specialty. Formally, the top-k policy that assures ϵ-privacy under Attack II is described below. Here, FP_s (TP_s) denotes the number of false (true) positive specialties. A false positive specialty is a disease that a patient does not have but there is at least one positive hospital that is specialized in.

Attack-II mitigation:

- **Top-k stop condition**:

$$FP_s \leq \frac{TP_s}{1 - \epsilon^{-1}} \tag{3}$$

- **Distance definition**:

$$\begin{aligned} D_{II}(N, \{P_i\}) &= \|S(N) \setminus \cup_i S(P_i)\| \\ &= \| \cap_i [S(N) \setminus S(P_i)]\| \end{aligned} \tag{4}$$

The distance D_{II} between a negative hospital N and a set of positive hospitals $\{P_i\}$ is defined in Equation 4. Here, $S(\cdot)$ denotes the vector of specialties of a hospital. We use hamming distance to capture the difference (\) between two specialty vectors. The distance definition favors the noises with different specialties from the true positive specialties. Thus, the number of false positive hospitals needed can be minimal, resulting in better search precision and performance.

Proposition: P^{3}I can mitigate Attack-II with assured ϵ-privacy.

Proof. From the top-k stop condition in Equation 3, we can have the following.

$$\begin{aligned} & FP_s \leq \frac{TP_s}{1 - \epsilon^{-1}} \\ \Rightarrow & \frac{FP_s}{FP_s + TP_s} \leq \epsilon \end{aligned} \tag{5}$$

Attack II follows the information flow $I, B \overset{a}{\mapsto} disease$ and is about recovering true positive specialties TP_s from the false positive ones FP_s. Then, the success rate is:
$Pr(disease|I, B) = \frac{FP_s}{FP_s+TP_s} \leq \epsilon$
Thus, the success rate is bounded by ϵ, hence ϵ-privacy. □

ϵ-Privacy under Attack III (Exact-match). We use the following top-k policy to mitigate Attack III with exact-match semantics.

Attack III mitigation (exact-match):

- **Top-k stop condition**:

$$FP \quad \geq \quad TP(\epsilon^{-1} - 1) \tag{6}$$

- **Distance definition**:

$$D_{III}(N, \{P_i\}) = \sum_i m_B(N, P_i) == 0?0 : \infty \tag{7}$$

[3] To be more specific, consider a counterexample against adopting l-diversity in P^{3}I. In Table 3 (case 1), 3-diversity, is already there without any noises. Because all true-positives have totally three specialties, that is, Cancer, Rehab, Woman's center. However, given no noises, the success rate of Attack II can be as high as 100%, leading to a situation that achieves 3-diversity yet no protection in the sense of ϵ-privacy.

Proposition: P^{3}I can mitigate Attack-III with assured ϵ-privacy in the sense of exact-match semantic.

Proof. Given Attack III follows the information flow $I, B \overset{a}{\mapsto} I^0$, the success rate is:

$$Pr(I^0|I,B) = \frac{TP}{TP + FP_0} \tag{8}$$

Here, we consider two types of false positive hospitals, the matching ones with zero-valued score (FP_0) and non-matching ones with one-valued score (FP_1). FP_1 can be distinguished by the adversary with background knowledge and thus should be discarded in accounting the success rate.

Our distance in Equation 7 is defined in such a way that any non-matching hospitals would lead to a distance of an infinitely large value, and thus will not be chosen by Algorithm 1. In other words, there are no non-matching hospitals that can be chosen as noises, that is,

$$\begin{aligned} & FP_1 = 0 \\ \Rightarrow \quad & FP = FP_0 + FP_1 = FP_0 \end{aligned} \tag{9}$$

Combining Equation 6, 8, 9, we can arrive at ϵ-privacy:

$Pr(I^0|I,B) \leq \epsilon$ □

ϵ-Privacy under Attack III (Fuzzy-match). We use the following top-k policy to mitigate Attack III with fuzzy-match semantics.

Attack III mitigation (fuzzy-match):

- **Top-k stop condition**:

$$\frac{\sum_{\forall\{TP\}} m_B(TP)}{\sum_{\forall P \in \{FP\} \cup \{TP\}} m_B(P)} \leq \epsilon \tag{10}$$

- **Distance definition**:

$$D_{III}(N, \{P_i\}) = \min_i m_B(N, P_i) \tag{11}$$

Under Attack III with fuzzy matching, what a rational adversary would do is to bias the attack towards positive hospitals with a small non-matching score. Specifically, we consider the adversary pick a positive hospital with probability inversely proportional to the non-matching score.[4] In the previous example about a Georgia patient, the adversary would avoid choosing the New York hospital due to its high non-matching score (or long geographic distance).

Proposition: P^{3}I can mitigate Attack-III with assured ϵ-privacy in the sense of fuzzy-match semantic.

[4]Another strategy is for an adversary to deterministically pick the hospital with the maximal score, which ignores all score information other than the maximal one.

Proof. To the rational adversary, the success rate can be modeled by Equation 12.[5]

$$Pr(I^0|I,B) \quad = \quad \frac{\sum_{\forall TP \in I^0} m_B(TP)}{\sum_{\forall P \in I} m_B(P)} \tag{12}$$

The intuition of Equation 12 can be best illustrated by the Georgia patient example. Assuming there are five hospitals in the P^{3}I search result, and the distances of the five hospitals to the patient home are $2.5, 2, 1, .25, .2$ as in Table 3. Considering case 2 where there are two true positives, h_2 and h_3, the success rate following the calculation in Equation 12 is $Pr(I^0|I,B) = \frac{2+1}{2.5+2+1+.25+.2}$.

Plugging Equation 10 into Equation 12, we arrive at the ϵ-privacy: $Pr(I^0|I,B) \leq \epsilon$. □

5. Secure Multi-Party Directory Construction with Optimization

In this section, we present the secure directory publication and the optimization techniques based on pre-computation. The general idea is to abstract the computation at different levels and precompute the computation at a specific level. This way, we present a series of precomputation techniques (in § 5.3, § 5.4) that vary in their aggressiveness. To start with, we present the naive approach based on multi-party computation (MPC) without precomputation. We first introduce the background on MPC.

5.1. Preliminary: Multi-Party Computation

In our protocol, we make use of existing multi-party computation (MPC) protocols whose background is presented here. In general, the purpose of MPC is to evaluate a function whose inputs are provided by different parties. Each input is private to its provider party. The protocol of MPC ensures that it does not leak any information about the private inputs even when the computation states are exchanged and shared. Different computational models exist in MPC, including circuit and RAM. After decades of studies, there are a variety of MPC protocols realizing different computation models, specialized for different network scales (for two, three or many parties). In particular, the protocol of GMW [34] is for multi-party, Boolean-circuit based MPC that is constructed based on the primitives of secret-sharing and oblivious transfers. The protocol of multi-server Private-Information Retrieval (ms-PIR) [36, 41] is a RAM-based MPC with multiple servers interacting a client on the computation of a simple selection operation (e.g., like a database selection).

[5]The basic assumption is that all true-positive hospitals have a non-matching score close to zero.

MPC causes high overhead, mainly due to the "data -oblivious" representation of the computation and cryptographic primitives being used in the construction. For more-than-three party computation, the use of secret sharing also cause high overhead as the shares need to be broadcast in the entire network. This unscalability (in data and network sizes) makes it challenging to apply MPC for real-world distributed applications.

In practice, the common way MPC is used for many-party distributed applications is based on the "outsourcing" paradigm. That is, given multiple input parties, the GMW protocol distributes the input shares to a small number of computing parties (e.g., three parties as in the Sharemind system [22]). The data security heavily relies on the non-collusion assumption of the computing parties. In our work, we deem this outsourcing model unsuitable for the target application. In HIPAA, a hospital cannot share patient data with *any* third-party entity without patient consent. Therefore, our problem considers each input party as computing party and the MPC protocol needs to run directly on a medium or large network.

5.2. MPC-based Publication

Privacy-preserving directory publication is an MPC problem as the input data are spread across multiple producers and are private to them. The naive way to realize directory publication is thus to place the computation as in List 1 into the MPC; this approach is denoted by M_0. Given the circuit representation of MPC program, the algorithm in List 1 can be easily converted to a circuit; the algorithm is a nested loop with pair-wise distance computation, and the data/control flow is essentially oblivious. In particular, we represent each producer by a vector (e.g., specialties of a hospital) and the similarity between producers can be realized by hamming distance. More complex string similarity computation is realized by dynamic-programming based algorithms which are also data oblivious. The security of this approach inherent from that of MPC.

This MPC approach is inefficient especially in big-data sharing scenario where there are a large amount of personal records. This is due to the expensive cryptographic primitives (e.g., oblivious transfers, etc) used in MPC protocols. To improve the performance, it relies on reducing the use of MPC in the distributed directory publication.

5.3. Full Precomputation Scheme

To reduce the use of MPC, we propose application-level precomputation. Given the $topK(T,S)$ algorithm in List 1 where only input T (the true producers) is private, we pre-compute the algorithm on the public input S and all possible values of private input T. The precomputation result is a table of results under different T values. Then, we use the actual value of T to privately look up this table and to securely retrieve the result entry. This stage can be realized in MPC using protocols such as multi-server private information retrieval (ms-PIR) [36, 41]. Formally, the full precomputation is to compute $topK(2^S, S)$ where 2^S is the power set of S which includes all possible values of private T. This scheme is named M_1.

The precomputation is effective in our directory-publication problem, provided the following characteristics. First the *topK* algorithm invokes some complex computation such as distance computation (i.e. Line 7 in List 1) which involve background knowledge about the producer profiles (e.g., hospital specialties and geographic locations). Precomputation avoids placing these complex computations in MPC which reduces overhead. Second, the precomputation only needs to be done once and its results can be *reused* for publishing different people's entries. Third, given the independence between different values, one can leverage data-parallelism to facilitate the computation. Note that the precomputation needs to be done for all possible value of T, that is, the power set of all producers; although the possibility combination grows exponentially with the number of producers, we only consider the data-producer network is moderately large. For instance, in healthcare, a regional or statewide HIE typically consists of less than hundreds of hospitals in a consortium.

The security of precomputation relies on the fact that no private value is involves in the precomputation. Private data only occurs in the actual MPC computation.

5.4. Selective Precomputation Schemes

The full precomputation scheme considers the directory computation of *topK* as a whole for precomputation. In this section, we dive into the computation *topK*() and *selectively* precompute certain computation-intensive parts in *topK*(). Concretely, our selective technique considers *topK* consists of distance-computation at different granularity. For one, it is to pre-compute the distance between T and $S-T$, considering all possible values of T. This way, we have the selective precomputation, M_2. For the other, it is to pre-compute the distance between all pairwise data producers. This yield the selective precomputation scheme, M_3.

In M_2, the precomputation considers all possible values of true-producer T. Given a value T^*, it precomputes the set-wise distance between T^* and $S-T^*$. This produces a distance table for the subsequent MPC. In the MPC, it first follows the computation in List 1 until Line 6. Then for Line 6 to 9, it is replaced by a secure lookup into the precomputation table. The lookup is realized by the ms-PIR protocol as in M_1.

In M_3, it precomputes the pair-wise distance matrix. That is, for any producer s_1 and $s_2 \in S$, it precomputes their distance and stores it in a table. Then, in the MPC stage, it follows the algorithm in List 1 except that the call to `dist(T[i],S[j])` is replaced by a ms-PIR lookup to the precomputation table.

The security of these precomputation schemes are straightforward, as all private-data related computations are placed inside the MPC/ms-PIR protocol whose security is proven. The precomputation only considers the public data.

In summary, the *topK* computation for privacy-preserving directory publication can be modeled as a process that issues a series of call to `dist(T[i],S[j])`. Our pre-computation schemes partitions this computation process at different "break" points and selectively places a certain partition to precomputation and the rest of computation into MPC/ms-PIR. Table 4 illustrates the three pre-computation schemes from this computation-partitioning perspective.

5.5. Data-Parallel Pre-Computation

The pre-computation handles multiple independent input values. There is innate data parallelism that can be exploited for better performing pre-computation. In our system, we realize it by data-parallel pre-computation tasks where each task with distinct input value runs in a dedicated thread. Different threads run concurrently and without synchronization. We implement this data-parallel pre-computation framework on both multi-core CPU and general-purpose GPU (GPGPU). Given the large number of possibilities in input values (and the simplicity of each task), GPGPU lends itself to the parallel pre-computation due to its scalable execution model.

In implementation, the CPU implementation is based on pthread library [13]. We pack multiple possible input values in one thread and the number of threads is twice the number of hyper-threads in hardware. The GPGPU implementation is based on CUDA library [2]. In this case, the underlying NVidia-Tesla GPU has global memory of 5 GB and threads run in one grid of 65,635 blocks, each of 1024 GPU threads. This architecture allows to scale the number of threads to 2^{27} and can easily handle the producer networks of more than 27 parties.

6. Evaluation

In this section, we study the feasibility of our technique for HIE applications in a holistic manner. Lacking benchmark dataset in existing literature, we first present a real healthcare dataset to populate the HIE data producers and locator. This sets up a target scenario for the performance study which we will present next. The purpose of performance evaluation is to answer the following question: *What is the overhead of privacy-preserving directory publication? and how effective is the proposed precomputation technique in performance optimization?*

6.1. Dataset

USNEWS dataset The USNEWS dataset [7] is used to model hospital profiles. The dataset considers 16 primary hospital-specialty categories, such as cardiology and rehabilitation (the entire list of specialties is shown in Table 6). For each category, a hospital is associated with a rating of three grades: "Nationally ranked", "High-performing", and "Null". We map "Nationally ranked" to value 2, "High-performing" to value 1, and "Null" (i.e. the hospital does not have the department for this specialty) to value 0. Each hospital is associated with other profile information, such as the resident city and state. Currently, we select the dataset to include 40 top-ranked hospitals (out of 180) in the New York metropolitan area.

Open-NY Health Dataset ("Sparcs") To model patient-wise hospital visits, we use an OPEN-NY dataset, called Sparcs [14]. The public dataset includes inpatient discharge records with identifiable information removed. At the finest granularity, it provides per-visit per-patient information (e.g., patient age group, gender, race, ethnicity and other de-identified information), the facility information (e.g., zip-code, name, service areas) and other per-visit information (e.g., admission type, the length of stay). Given the identifiable patient information is removed, we model the per-patient visit history by aggregating the records based on available quasi-identity information (i.e. age group, race, ethnicity, etc).

6.2. Protection Effectiveness

In our security analysis, we consider a probabilistic attacker and the ϵ-privacy assurance, which come with two limitations: one is that it only considers attacks against one specific patient, and the other is that ϵ-privacy provides assurance in a statistical sense. To complement the security analysis, we move forward to measure the *variance* of success rate in a broader sense, that is, considering all patients.

Given the flexibility that the attacker now has in choosing which patient to attack, we consider the attacker can naturally exhaust all her options and target on the most vulnerable patient. The attacker can gauge the vulnerability of a patient by various

Table 4. Partitioning $topK$ algorithm to the precomputation-MPC framework: For notation in this table, T, S are true and all producers as in the $topK()$ algorithm in List 1. D_i for $i = 1, 2, 3$ are the table storing precomputation results. MPC is secure multi-party computation protocol and $msPIR$ is a special MPC protocol for multi-server private information retrieval.

	Pre-compute	MPC+msPIR
M_0	-	$topK(T, S)$
M_1	$D_1 = topK(2^S, S)$	$Lookup_{msPIR}(D_1, T)$
M_2	$D_2 = dist(2^S, S)$	$topK2_{MPC}(T, S)$ invoking $Lookup_{msPIR}(D_2, T)$
M_3	$D_3 = dist(S, S)$	$topK3_{MPC}(T, S)$ invoking $Lookup_{msPIR}(D_3, T[i], S[j])$

Table 5. Specialty catalog in the USNEWS dataset

Table 6. Specialty catalog in the USNEWS dataset

Index	Name	Index	Name
0	Cancer	8	Neurology & Neurosurgery
1	Cardiology & Heart Surgery	9	Ophthalmology
2	Diabetes & Endocrinology	10	Orthopedic
3	Ear, Nose & Throat	11	Psychiatry
4	Gastroenterology & GI Surgery	12	Pulmonology
5	Geriatrics	13	Rehabilitation
6	Gynecology	14	Rheumatology
7	Nephrology	15	Urology

Table 7. Experiment platform

New York Server	
CPU	Xeon(R) E5-2640 v3 @ 2.60GHz
	2 processors/16 cores/32 hyper-threads
Memory	245 GB

California Server	
CPU	Xeon(R) E5-2687W @ 3.10GHz
	2 processors/16 cores/32 hyper-threads
Memory	256 GB
GPGPU	Nvidia Tesla K20c
	1 grid/65535 blocks/2^{27} threads
	Global Memory 5119MB

metrics, such as, the one with the smallest number of specialties in her positive hospitals. Then, given the P^{3}I is configured with a user-defined ϵ, we measure the actual success rates (of attacks on vulnerable patients) and report those that are larger than $epsilon$. In the experiment, we consider $epsilon = 0.4$ and 0.5. Note that we deliberately avoid to use average success rate (by multiple patients), since it is the largest success rate that makes the system vulnerable.

Overall effectiveness. We compare the case of P^{3}I with alternatives, including no-protection and grouping-based PPI. "No-protection" is the baseline which publishes raw location meta-data (i.e., patient-to-hospital information) without any noises. Grouping PPIs [19, 61] are based on the idea of K-anonymity (note to avoid confusion, we use K for K-anonymity, and k for top-k), which works by randomly grouping K hospitals together. We present our study results in Table 8. Here, for P^{3}I, we use the policy of adaptively choosing top-k to achieve a constant diversity l. To make the comparison fair, we use the same budget for injecting noises; that is, the amount of total false positives in P^{3}I is kept the same to that in grouping-based PPI. In the table, it is clear that P^{3}I achieves significantly smaller attack success rate and number of incidents.

Table 8. Effectiveness of P^{3}I

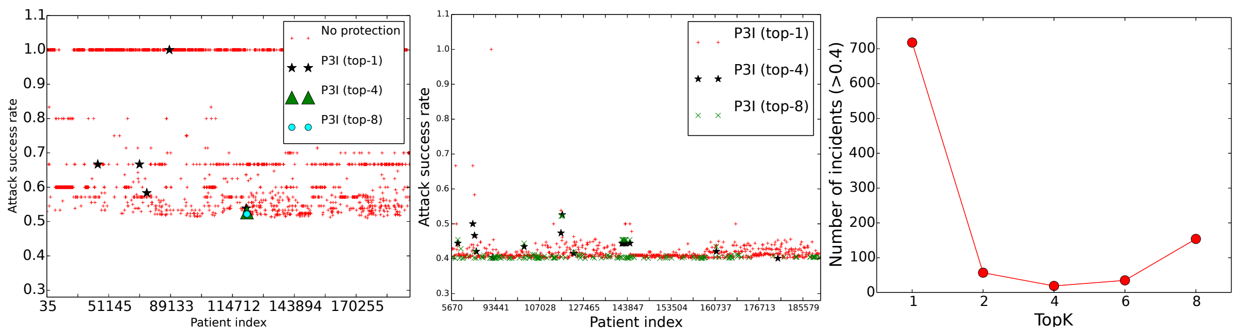
	Avg. success rate (> .5)	No. of incidents (> .5)
P^{3}I	0.537651	14
No-protection (Broadcasting)	0.782902	2349
PPIs [19, 61]	0.68231	1298

Effectiveness of top-k algorithm. We first report all incidents with success rates higher than the user-defined $\epsilon = 0.5$. The results are reported in Figure 3a where the x axis is the index of patients (in our processed health dataset, there are totally 280, 000 patients). It is easy to see that the no-protection approach results in much more densely distributed dots than P^{3}I under various configurations of k. Furthermore, it is often the case that no-protection results in 100% success rate, implying the real-world dataset is vulnerable to probabilistic attacks when without protection. This result is consistent to Table 8 and explains the difference there.

We then manually vary the value of k to measure its effect on the attack success rate. The experiment result is presented in Figure 3b and 3c. It is interesting to see that it is not always the case that setting a larger k results in better protection; the protection in terms of larger-than-configured-ϵ incident rate is minimized at $k = 6$. Our preliminary inspection shows that this is relevant to the fact that real-world dataset is erroneous and does not fully match with some of our assumptions (e.g., patient does not always go to the nearest hospitals).

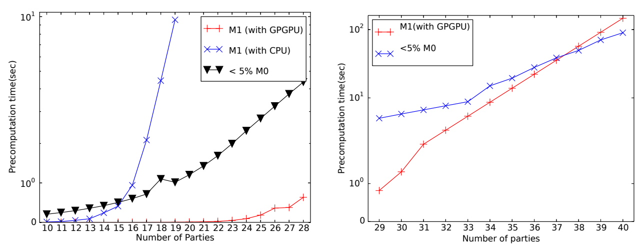
6.3. Performance of Directory Publication

We first conduct micro-benchmark to test the performance of data-parallel precomputation. Then, we test the overall performance of secure directory publication, with a machine of multi-core processor and in a geo-distributed setting.



(a) P^{3}I protection vs no protection (b) Varying k in top-k algorithms (c) Number of incidents (> 0.4) with varying k

Figure 3. Success rates of attacks



(a) With ≤ 28 parties (b) With > 28 parties

Figure 4. Pre-computation performance with GPGPU

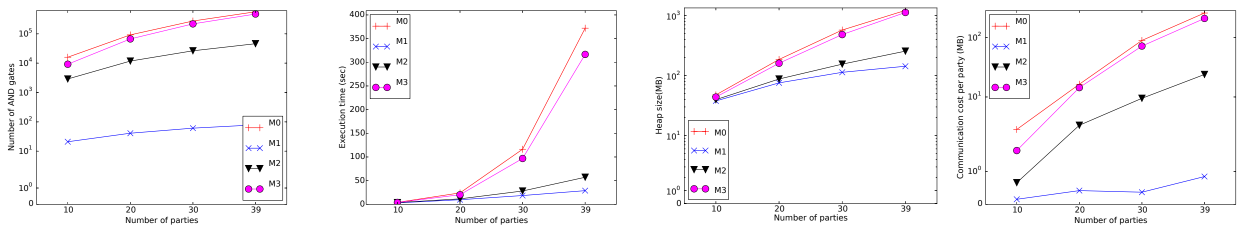
Micro-benchmark of Pre-computation. The pre-computation is implemented with data parallelism (as described in 5.5) and runs on multi-core CPU and GPGPU. We report the time to pre-compute on GPGPU and that on CPU in Figure 4. This figure also includes a baseline which is the 5% execution time of running M_0 (i.e. without any precomputation).

The performance result in Figure 4a shows that GPGPU based pre-computation is effective in reducing the execution time, and its overhead is negligible comparing the baseline. Concretely, the CPU based precomputation has its execution time to quickly surpass the baseline when the network grows over 15 parties. The GPGPU-based precomputation has much lower overhead than the baseline for any network with less than 28 parties.

For more than 28 parties, all GPU threads are occupied and it will need multiple iterations in transferring data from GPU's global memory to host memory. As a result, the GPGPU precomputation time increases exponentially, also reported in the Figure 4b (note that the y axis is of log scale). With a single GPGPU card, the precomputation time surpasses the baseline when the network is larger than about 40 parties. Here, we stress that the typical scale of a healthcare consortium is usually medium-sized (e.g., tens of hospitals and clinical centers). For nation-wide healthcare systems, there may be thousands of hospitals. In this case, one can use more GPGPU cards to do the precomputation in parallel, while retaining the efficiency.

Overall Performance with MPC. **The MPC-based implementation** of directory publication is realized on the GMW software [24], an open-source MPC software and Percy++ [12], an open-source multi-server PIR software. We note that our precomputation protocol only relies on the general MPC and PIR interface and other MPC "backend" software can be used in our protocol. The GMW protocol exposes a circuit-based programming interface that requires MPC programmers to write a generator for Boolean circuit encapsulating the intended computation logic. At runtime, the GMW protocol runs on multiple parties where each party generates and executes the circuit by iterating through all gates in the circuit (following a topologically sorted order); for each gate, the evaluation is synchronized across all parties. The GMW protocol makes bit-wise use of two cryptographic primitives which provides the security of the protocol, that is, secret sharing [57] and oblivious transfer [54]. In particular, the per-gate evaluation in GMW is to broadcast the shares of input-wire bit to all the parties in the entire network. In our application, we manually express the logic of *topK* algorithm in the GMW Boolean circuit, and tightly estimate the number of gates to pre-allocate so that the unused GMW circuit can be optimized out. Our GMW-based implementation consists of about 1500 lines of C++ code.

Multi-processing execution platform: We first run our protocol on a single node with multi-processing.



(**a**) Number of AND gates (**b**) Running time (**c**) Heap size per party (**d**) Comm. cost per party

Figure 5. Performance of directory publication based on precomputation and MPC

The machine specs are in Table 7 (the New York server). In this setting, each process represents a data producer and runs a GMW party. In the execution, each process holds a dedicated copy of the entire circuit allocated in its virtual-memory space and without shared memory. The machine has memory large enough (245 GB in total) to hold all circuit copies of the 39 parties without paging.

Results on multi-processing: To measure the performance of MPC, we used four metrics, the number of AND gates (1), end-to-end execution time (2), memory consumption (3) and communication costs (4). 1) We report the number of AND gates in the compiled GMW Boolean circuit. This metric helps evaluate the performance in a hardware-independent fashion. We only consider AND gates in a circuit and ignore other gates (i.e., XOR gates) because evaluating XOR is free (i.e. free-XOR technique [25]) and evaluating AND gates dominates the cost. 2) We report the wall-clock time from launching the first process to the completion of the last process. 3) We report the size of the heap memory in GMW that stores all circuit gates. It is measured by the Valgrind framework (particularly the Massif memory profiler [56]). 4) We report the party-to-party communication overhead, by monitoring all outbound messages through the socket port of each process using IPTraf[6].

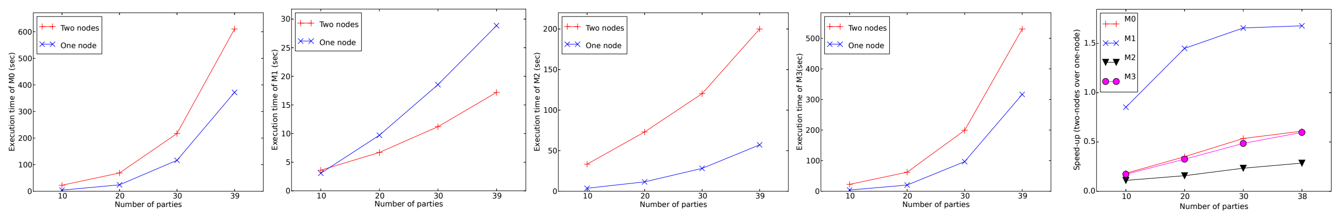
In the experiment, we vary the number of parties (or data producers) and present the result in Figure 5. Figure 5a reports the result of AND gate number and Figure 5b reports wall-clock running time. They both show that the pre-computation based schemes (i.e. M_1, M_2, M_3) outperform the baseline without pre-computation. Notably, the M_1 scheme causes the best performance with a speedup of 13 times (comparing the baseline M_0) in the setting of 39 parties. This result demonstrates the effectiveness of pre-computation techniques that off-loads computation from the expensive MPC. In terms of memory consumption in Figure 5c, M_1 and M_2 are close, reducing up to memory consumption roughly by an order of magnitude comparing M_0 and M_3. It shows that while M_1 produces pre-computation results as additional data, its much smaller circuit (for simple lookup operation in ms-PIR) makes the overall saving of memory footprint as compared to the baseline M_0. In Figure 5d, the communication overhead of M_1 stays to be the smallest among the four schemes, with a saving of more than 2 orders of magnitudes comparing M_0. This is consistent with the result in the number of AND gates.

Geo-distributed execution platform: We conduct the experiment with two servers set apart more than 3000 miles (one server in the State of New York, and the other in the State of California). The bandwidth is 100 Mbps. The specification of the two servers is illustrated in Table 7. Each server runs half of the parties with multiprocessing. Different parties communicate through sockets. The precomputation runs only in one server.

Results with geo-distributed execution report the execution time of the four schemes in the geo-distributed setting. The results are in Figure 6. For comparison, we include the results in the single-node setting. The execution time grows super-linearly with the number of parties in a network. For M_0, M_2 and M_3, running them on two geo-distributed nodes leads to longer execution time. Interestingly for M_1, the geo-distributed execution is faster than the single-node one. In this case, the performance slowdown caused by the slower communication channels is offset by the performance gain from the extra hardware (e.g., CPU) on multiple nodes. We suspect this performance result is due to that the MPC is dominated more by the local computations (on secret shares) and less by the network communications.

7. Related Work

[6]http://iptraf.seul.org/



(a) Baseline M_0 **(b)** Precomputation M_1 **(c)** Precomputation M_2 **(d)** Precomputation M_3 **(e)** Speedup

Figure 6. Geo-distributed performance on the Internet

7.1. Privacy-Preserving Data Federation

Multi-party noise generation Distributed differential privacy [30, 53, 63] is proposed to support privacy-preserving aggregations. The randomized response [63] provides differential privacy yet with uncontrollable noises and loss of utility. PrivaDA [32] is proposed to achieve the optimal utility and performance optimization by adopting arithmetic circuit based MPC for the noise generation. Existing multi-party noise generation takes a randomized approach and mainly targets for statistical aggregation (e.g., distributed differential privacy). This is inapplicable to our problem which features deterministic noise generation for the rigorous privacy guarantee, and needs to serve non-aggregation queries.

PPI Privacy-Preserving Index or PPI is proposed to federate and index distributed access-controlled documents [18, 19] and databases (e.g., patient medical records in the HIE locator service) [61] among autonomous providers. Being stored on an untrusted server, PPI entails preserving the content privacy of all participant providers or hospitals. Inspired by the privacy definition of K-anonymity [59], existing PPI work [18, 19, 61] follows the *grouping-based* approach; it organizes providers into disjoint privacy groups of size K, such that providers from the same group are indistinguishable. However, K-anonymity, while easy to construct, does not guarantee high-quality privacy preservation. In addition, early approaches of PPI construction [47, 60] are based on randomized responses [63], an iterative protocol that takes indefinite number of rounds to converge and may produce incorrect result (with certain probability). To avoid those drawbacks, ϵ-PPI combines randomized responses with a minimal use of multi-party computation to construct PPI correctly and efficiently.

7.2. MPC Frameworks and Optimization

In the last decade, practical MPC has attracted a large body of research work with a focus on programming language support and optimization [16, 20–22, 24, 37, 43, 50, 55]. Practical MPCs are built on top of cryptographic protocols, such as Yao's garbled circuits [64] or GMW protocol [34], with protocol-level optimization, such as Oblivious Transfer (OT) extensions [38], or for stronger security, such as resilience with dishonest majority [27]. The MPC protocols assume a circuit interface to express the computation, and practical programming support focuses on compiling a program written in a high-level language into the circuit. Existing MPC protocols and systems mainly focus on a small-scale computing that involves 2 or 3 parties. To the general MPC problem, a fundamental trade-off exists between performance and computation generality; for instance, randomized responses [63] and other techniques for privacy-preserving data mining take an ad-hoc and domain-specific approach, which can be efficient at scale. By contrast, the general-purpose MPC is rather expensive.

MPC Optimization High performance overhead stays to be one of the major hurdles to applying MPC in practice, which is partly caused by MPC's fine-grained use (e.g., per single bit) of expensive cryptographic primitives, and the need to transfer all *possible* computation results for the "obliviousness" of computation flow. Various optimization techniques are proposed to utilize the programming semantics to reduce the circuit size and depth (e.g., by using the hardware synthesis tools [28, 58]) and optimize the resource utilization (e.g., just-in-time compilation and pipelined execution [37, 43]). Program analysis [42] is used to automatically infer privacy-sensitive data and constraints MPC only to the sensitive data. [44] conducts pre-processing on verification of MPC and results in general transformation from a passively secure protocol to an actively secure one. Our MPC optimization is currently specific to the directory construction problem, while holding the potential to apply to more generic computations.

Some programming frameworks support high-level programming languages with compilers (e.g., Fairplay(MP) for SFDL [21, 50], Sharemind for SecreC [39], CBMC-GC for ANSI C [33], PCF for C [43], Wysteria for a high-level typed specification language [55], PICCO for C with extension [66]), while others expose a quite low-level circuit based interface (e.g., GMW [24], JustGarble [20], OTExtension [16]); particularly both boolean circuit (e.g., GMW) and arithmetic circuit (e.g., SEPIA [23]) are considered. In addition, some advanced technique designs based on hybrid model that combines both boolean or arithmetic circuits (e.g., ABY [29], TASTY [35], Wysteria [55]).

7.3. Anonymization Definitions

Publishing public-use data about individuals without revealing sensitive information has received a lot of research attentions in the last decade. Various anonymization definitions have been proposed and gained popularity, including K-anonymity [59], l-diversity [48], t-closeness [45], and differential privacy [31]. In addition, prior work [51] formally studied the information leakage under background knowledge attacks by formulating the problem using a proposed declarative language. These anonymity notions however are generally inapplicable to the PPI problem – they are mainly designed for statistic analysis or aggregation style computation where the result is global pertable data, while PPI needs to serve queries specific to individual records.

r-confidentiality [65] is a privacy notion specific to the PPI problem. It assumes a probabilistic attacker on PPI and considers the increase of attack success-rate with/without using the background knowledge. By contrast, our proposed ϵ-privacy considers to bound the attack success-rate (instead of the increase) which we believe provides better privacy control.

8. Conclusion

This work presents an MPC-precomputation framework tailored for privacy-preserving data publication for data-sharing applications. The pre-computation framework improves the performance by minimizing the private-data computation and realizing the public-data only pre-computation in a data-parallel fashion. Several pre-computation policies are proposed with varying degrees on the aggressiveness. It is demonstrated that the proposed pre-computation scheme is applicable in real health-care scenarios. Based on real datasets and implementation on open-source MPC software, the performance study shows that the proposed pre-computation achieves a speedup of more than an order of magnitude without security loss.

References

[1] CommonWell: http://www.commonwellalliance.org/.
[2] Cuda: https://en.wikipedia.org/wiki/cuda.
[3] Directive 95/46/ec of the european parliament and of the council.
[4] GaHIN: http://www.gahin.org/.
[5] HealthEConnections: http://www.healtheconnections.org/rhio.
[6] HiPAA, http://www.cms.hhs.gov/hipaageninfo/.
[7] http://health.usnews.com/best-hospitals/area/new-york-ny/specialty.
[8] NHIN Connect, http://www.connectopensource.org/.
[9] NHIN: https://www.healthit.gov.
[10] Ohio voter files: https://www6.sos.state.oh.us/ords/f?p=111:1.
[11] OpenEMPI: http://www.openempi.org/.
[12] Percy++/pir in c++: http://percy.sourceforge.net/.
[13] pthread: https://en.wikipedia.org/wiki/posix_threads.
[14] SPARCS: http://www.health.ny.gov/statistics/sparcs/.
[15] *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015.
[16] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 535–548, 2013.
[17] J. Bater, G. Elliott, C. Eggen, S. Goel, A. N. Kho, and J. Duggan. SMCQL: secure query processing for private data networks. *CoRR*, abs/1606.06808, 2016.
[18] 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.
[19] M. Bawa, R. J. B. Jr., and R. Agrawal. Privacy-preserving indexing of documents on the network. In *VLDB*, pages 922–933, 2003.
[20] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 478–492. IEEE Computer Society, 2013.
[21] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp: a system for secure multi-party computation. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM Conference on Computer and Communications Security*, pages 257–266. ACM, 2008.
[22] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, pages 192–206, 2008.
[23] M. Burkhart, M. Strasser, D. Many, and X. A. Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 223–240. USENIX Association, 2010.
[24] S. G. Choi, K. Hwang, J. Katz, T. Malkin, and D. Rubenstein. Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. In O. Dunkelman, editor, *Topics in*

Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings, volume 7178 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2012.

[25] S. G. Choi, J. Katz, R. Kumaresan, and H.-S. Zhou. On the security of the free-xor technique. In *Theory of Cryptography*, pages 39–53. Springer, 2012.

[26] R. Cramer, I. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[27] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In J. Crampton, S. Jajodia, and K. Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[28] D. Demmler, G. Dessouky, F. Koushanfar, A. Sadeghi, T. Schneider, and S. Zeitouni. Automated synthesis of optimized circuits for secure computation. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1504–1517. ACM, 2015.

[29] D. Demmler, T. Schneider, and M. Zohner. Aby - a framework for efficient mixed-protocol secure two-party computation. In *Network and Distributed System Security Symposium (NDSS'15)*, Feb. 2015.

[30] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[31] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[32] F. Eigner, M. Maffei, I. Pryvalov, F. Pampaloni, and A. Kate. Differentially private data aggregation with optimal utility. In *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014, New Orleans, LA, USA, December 8-12, 2014*, pages 316–325, 2014.

[33] M. Franz, A. Holzer, S. Katzenbeisser, C. Schallhart, and H. Veith. CBMC-GC: an ANSI C compiler for secure two-party computations. In A. Cohen, editor, *Compiler Construction - 23rd International Conference, CC 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, volume 8409 of *Lecture Notes in Computer Science*, pages 244–249. Springer, 2014.

[34] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In A. V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[35] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Tasty: tool for automating secure two-party computations. In *ACM CCS*, pages 451–462, 2010.

[36] R. Henry, F. G. Olumofin, and I. Goldberg. Practical PIR for electronic commerce. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 677–690, 2011.

[37] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*. USENIX Association, 2011.

[38] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 145–161, 2003.

[39] R. Jagomägis. Secrec: a privacy-aware programming language with applications in data mining.

[40] P. Jurczyk, J. J. Lu, L. Xiong, J. D. Cragan, and A. Correa. FRIL: A tool for comparative record linkage. In *AMIA 2008, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2008*, 2008.

[41] M. Keller and P. Scholl. Efficient, oblivious data structures for MPC. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 506–525, 2014.

[42] F. Kerschbaum. Automatically optimizing secure computation. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 703–714, 2011.

[43] B. Kreuter, A. Shelat, B. Mood, and K. R. B. Butler. PCF: A portable circuit format for scalable two-party secure computation. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 321–336, 2013.

[44] P. Laud and A. Pankova. Preprocessing-based verification of multiparty protocols with honest majority. *IACR Cryptology ePrint Archive*, 2015:674, 2015.

[45] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 106–115, 2007.

[46] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. Oblivm: A programming framework for secure computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015* [15], pages 359–376.

[47] Y. T. L. Liu. Privacy-preserving multi-keyword search in information networks. *IEEE Trans. Knowl. Data Eng.*, 27(9):2424–2437, 2015.

[48] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[49] A. B. Makulilo. *Asian Data Privacy Laws, Trade and Human Rights Perspective*, by graham greenleaf. *I. J. Law and Information Technology*, 23(3):322–324, 2015.

[50] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - secure two-party computation system. In M. Blaze, editor, *USENIX Security Symposium*, pages 287–302. USENIX, 2004.

[51] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern. Worst-case background knowledge for privacy-preserving data publishing. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 126–135, 2007.

[52] S. McCamant and M. D. Ernst. Quantitative information flow as network flow capacity. In *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 193–205, 2008.

[53] M. Pettai and P. Laud. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, December 7-11, 2015*, pages 421–430, 2015.

[54] M. O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[55] A. Rastogi, M. A. Hammer, and M. Hicks. Wysteria: A programming language for generic, mixed-mode multiparty computations. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 655–670. IEEE Computer Society, 2014.

[56] J. Seward, N. Nethercote, and J. Weidendorfer. *Valgrind 3.3-Advanced Debugging and Profiling for GNU/Linux applications*. Network Theory Ltd., 2008.

[57] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[58] E. M. Songhori, S. U. Hussain, A. Sadeghi, T. Schneider, and F. Koushanfar. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015* [15], pages 411–428.

[59] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[60] Y. Tang, L. Liu, A. Iyengar, K. Lee, and Q. Zhang. e-ppi: Locator service in information networks with personalized privacy preservation. In *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pages 186–197, 2014.

[61] Y. Tang, T. Wang, and L. Liu. Privacy preserving indexing for ehealth information networks. In *CIKM*, pages 905–914, 2011.

[62] C. Toth, E. Durham, M. Kantarcioglu, Y. Xue, and B. Malin. Soempi: A secure open enterprise master patient index software toolkit for private record linkage. In *AMIA Annual Symposium Proceedings*, volume 2014, page 1105. American Medical Informatics Association, 2014.

[63] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[64] A. C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

[65] 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.

[66] Y. Zhang, A. Steele, and M. Blanton. PICCO: a general-purpose compiler for private distributed computation. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 813–826, 2013.