Yigzaw et al. BMC Medical Informatics and Decision Making (2017) 17:1
DOI 10.1186/s12911-016-0389-x

BMC Medical Informatics and Decision Making

TECHNICAL ADVANCE Open Access

Secure and scalable deduplication of horizontally partitioned health data for privacy-preserving distributed statistical computation

Kassaye Yitbarek Yigzaw[1,2*], Antonis Michalas[3] and Johan Gustav Bellika[2,4]

Abstract

Background: Techniques have been developed to compute statistics on distributed datasets without revealing private information except the statistical results. However, duplicate records in a distributed dataset may lead to incorrect statistical results. Therefore, to increase the accuracy of the statistical analysis of a distributed dataset, secure deduplication is an important preprocessing step.

Methods: We designed a secure protocol for the deduplication of horizontally partitioned datasets with deterministic record linkage algorithms. We provided a formal security analysis of the protocol in the presence of semi-honest adversaries. The protocol was implemented and deployed across three microbiology laboratories located in Norway, and we ran experiments on the datasets in which the number of records for each laboratory varied. Experiments were also performed on simulated microbiology datasets and data custodians connected through a local area network.

Results: The security analysis demonstrated that the protocol protects the privacy of individuals and data custodians under a semi-honest adversarial model. More precisely, the protocol remains secure with the collusion of up to $N-2$ corrupt data custodians. The total runtime for the protocol scales linearly with the addition of data custodians and records. One million simulated records distributed across 20 data custodians were deduplicated within 45 s. The experimental results showed that the protocol is more efficient and scalable than previous protocols for the same problem.

Conclusions: The proposed deduplication protocol is efficient and scalable for practical uses while protecting the privacy of patients and data custodians.

Keywords: Bloom Filter, Data Reuse, Deduplication, Distributed Statistical Computation, Data Linkage, Duplicate Record, Electronic Health Record, Privacy, Record Linkage, Set Intersection

Background

Electronic health record (EHR) systems have been in existence for many years. The increased adoption of EHR systems has led, and continues to lead, to the collection of large amounts of health data [1]. Large amounts of administrative, survey, and registry data are also being collected. These data could aid in the development of scientific evidence that helps improve the effectiveness, efficiency, and quality of care of healthcare systems [2–4].

Introduction

The focus of this paper is the reuse of health data horizontally partitioned between data custodians, such that each data custodian provides the same attributes for a set of patients. Reusing data from multiple data custodians provides a sufficient number of patients who satisfy the inclusion criteria of a particular study. The number of patients at a single data custodian may provide insufficient statistical power, especially for studies on rare

* Correspondence: kassaye.y.yigzaw@uit.no
[1] Department of Computer Science, UiT The Arctic University of Norway, 9037 Tromsø, Norway
[2] Norwegian Centre for E-health Research, University Hospital of North Norway, 9019 Tromsø, Norway
Full list of author information is available at the end of the article

exposures or outcomes. When data are collected across multiple data custodians, the data of a heterogeneous mix of patients can be reused.

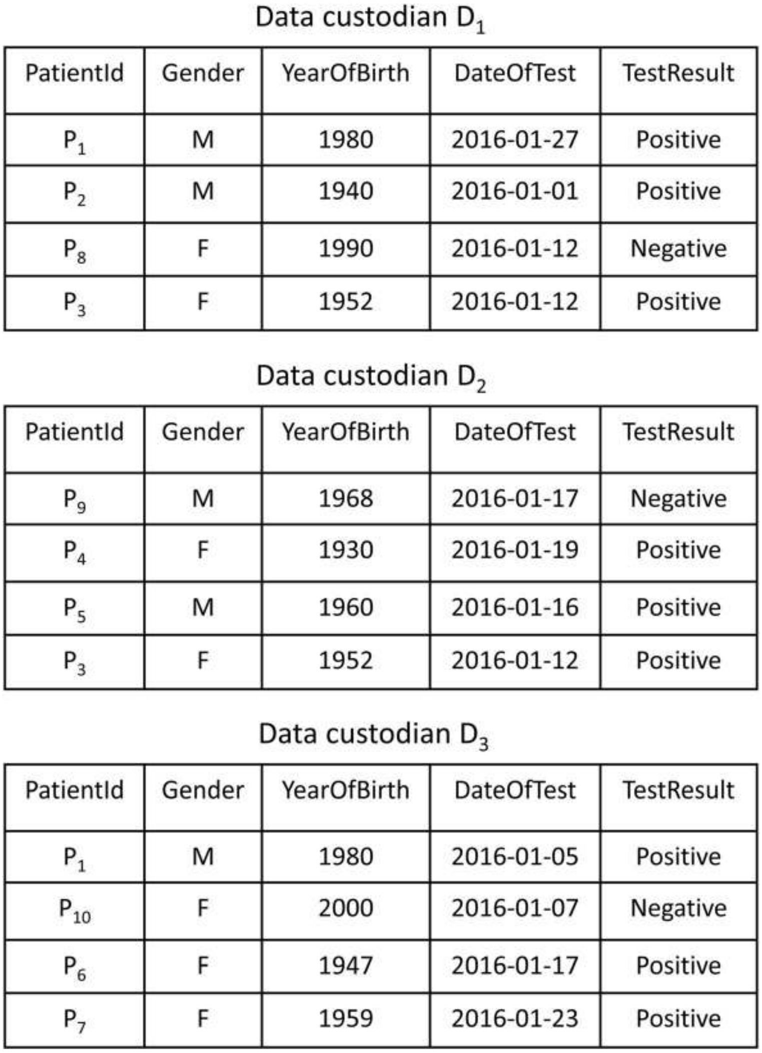
There has been substantial interest in the reuse of EHR data for public health surveillance, which also requires data from multiple data custodians covering the geographic area of interest [5–7]. One of the EHR meaningful use criteria in the United States is the ability to release health data for public health surveillance [8].

The horizontally partitioned datasets required for a health study or disease surveillance are often queried by distributing the query to data custodians, who execute the query and store a copy of the data extracts locally [9]. We refer to the data extracts distributed across data custodians as a virtual dataset (VD). Consider the execution of the query "select the records of patients tested for influenza A viruses in January 2016" across three data custodians $\mathcal{D} = \{D_1, , D_2, , D_3\}$. Figure 1 illustrates a VD that consists of the query results for the data custodians.

A VD may contain duplicate records from data custodians that cover overlapping areas and areas in close proximity [10–12]. The duplicate records can be exact or approximate. A set of records are exact duplicates if the records are compared using exact comparison functions, and the records have the same value for all attributes used for comparison. In contrast, approximate duplicate records are compared using comparison functions that allow approximate similarities, and the records have different values for one or more attributes.

Privacy-preserving distributed statistical computation

Access to and the use of patient data for research raise significant privacy concerns for the various stakeholders (i.e., patients and data custodians) [7, 13, 14]. A recent



Data custodian D_1

PatientId	Gender	YearOfBirth	DateOfTest	TestResult
P_1	M	1980	2016-01-27	Positive
P_2	M	1940	2016-01-01	Positive
P_8	F	1990	2016-01-12	Negative
P_3	F	1952	2016-01-12	Positive

Data custodian D_2

PatientId	Gender	YearOfBirth	DateOfTest	TestResult
P_9	M	1968	2016-01-17	Negative
P_4	F	1930	2016-01-19	Positive
P_5	M	1960	2016-01-16	Positive
P_3	F	1952	2016-01-12	Positive

Data custodian D_3

PatientId	Gender	YearOfBirth	DateOfTest	TestResult
P_1	M	1980	2016-01-05	Positive
P_{10}	F	2000	2016-01-07	Negative
P_6	F	1947	2016-01-17	Positive
P_7	F	1959	2016-01-23	Positive

Fig. 1 A simplified virtual dataset of influenza A test results distributed across three data custodians

approach that addresses privacy concerns is secure multiparty computation (SMC), which deals with the problem of computing a function f on distributed data without revealing any information except the results. SMC protocols have been developed for the statistical computation of distributed data that do not reveal anything except the results [15–21].

Statistical analysis of a virtual dataset that contains duplicate records may lead to incorrect results. Let us consider a query of the number of patients in a VD that satisfy a set of criteria. When there are duplicate records in the VD, a simple summation of the data custodians' local counts will not return the same result if the query is run against the combined datasets of all data custodians stored in a central database.

For example, the distributed statistical computation of the number of women who tested positive for influenza A against the VD shown in Fig. 1 would return an incorrect result. Patient P_3 would be counted twice, as she has a positive test result stored in D_1 and D_2. Therefore, to improve the accuracy of the statistical results, deduplication of the VD is a necessary preprocessing step before statistical analysis is conducted.

Privacy-preserving deduplication

Deduplication (also known as record linkage) is the process of linking records at the same or different data custodians that refer to the same individual. In contrast to record linkage, the final goal of deduplication is to remove duplicate records while maintaining a single occurrence of each record. Privacy-preserving record linkage (PPRL; also known as private set intersection and private record linkage) protocols have been developed to link records across multiple data custodians without revealing any information other than the linkage result [22, 23]. The main challenges of these protocols for practical use include the quality of the linkage, privacy, efficiency, and scalability [22].

The objective of this paper is to develop an efficient and scalable protocol for the deduplication of a VD while protecting the privacy of the patients and the data custodians. The proposed protocol supports various deterministic record linkage algorithms.

Our main contributions can be summarized as follows: We propose a novel efficient and scalable protocol based on Bloom filters for the privacy-preserving deduplication of a horizontally partitioned dataset. We provide proof of the security of the protocol against a semi-honest adversarial model in which the participating entities are assumed to follow the protocol steps, but the entities may try to learn private information from the messages exchanged during the protocol execution. We conducted a theoretical analysis of the protocol's efficiency and scalability. We implemented a prototype of the protocol and ran experiments among three microbiology laboratories located in Norway. We also ran experiments using simulated microbiology laboratory datasets with up to 20 data custodians and one million records.

The remainder of this section presents a review of related work and provides a use case for the deduplication problem and formally presents it. The Methods section outlines the requirements of the proposed protocol, as well as the threat model and assumptions, Bloom filter, notations, basic set operations, and secure sum protocol used in the protocol. Then, the proposed protocol is described. The Results section presents the security analysis, implementation, and evaluations of the protocol. Finally, the Discussion and Conclusions are presented.

Related work

Several PPRL protocols have been developed based on either deterministic or probabilistic matching of a set of identifiers. Interested readers are referred to [22, 23] for an extensive review of the PPRL protocols. The protocols can be broadly classified as protocols with or without a third party. In this section, we review privacy-preserving protocols for deterministic record linkage. These protocols are secure against the semi-honest adversarial model, which is the adversarial model considered in this paper.

A record contains a set of identifiers that consists of direct and indirect (quasi-identifier) identifiers and other health information. Direct identifiers are attributes that can uniquely identify an individual across data custodians, such as a national identification number (ID). In contrast, quasi-identifiers are attributes that in combination with other attributes can identify an individual, such as name, sex, date of birth, and address. In this paper, the terms identifier and quasi-identifier are used interchangeably.

Weber [12] and Quantin et al. [24] proposed protocols that use keyed hash functions. These protocols require data custodians send a hash of their records' identifiers to a third party that performs exact matching and returns the results. The data custodians use a keyed hash function with a common secret key to prevent dictionary attacks by the third party. These protocols are secure as long as the third party does not collude with a data custodian. Quantin et al.'s protocol [24] performs phonetic encoding of the identifiers (i.e., last name, first name, date of birth, and sex) before hashing, in order to reduce the impact of typing errors in the identifiers on the quality of the linkage.

Several protocols were proposed based on commutative encryption schemes[1] [25–27]. In these protocols, each data custodian, in turn, encrypts the unique identifiers for all records across the data custodians using its private key, and consequently, each unique identifier is encrypted with the private keys of all the data custodians. Then, the encrypted unique identifiers are compared with each other, as the encrypted values of two unique identifiers match if the two unique identifiers match. The protocols proposed in [25, 26] are two-party computation protocols, whereas Adam et al.'s [27] protocol is a multi-party computation protocol.

The protocols reviewed thus far require the exchange of a long list of hash or encrypted identifiers, which can limit the scalability of the protocols as the number of data custodians and records increases. In addition, protocols based on commutative encryption require communication rounds quadratic with the number of data custodians.

Multi-party private set intersection protocols were designed based on Bloom filters[2] [28, 29]. In general, each data custodian encodes the unique identifier values of its records as a Bloom filter (see the description of a Bloom filter in the Methods section). The protocols use different privacy-preserving techniques, as discussed below, to intersect the Bloom filters and then create a Bloom filter that encodes the unique identifiers of the records that have exact matches at all data custodians. Then, the data custodian queries the unique identifiers of its records in the intersection Bloom filter to identify the records that match.

In Lai et al.'s [28] protocol, each data custodian splits its Bloom filter into multiple segments and distributes them to the other participating data custodians while keeping one segment for itself. Then, each data custodian locally intersects its share of the Bloom filter segments and distributes it to the other data custodians. Finally, the data custodians combine the results of the intersection of the Bloom filter segments to create a Bloom filter that is an intersection between all the data custodians' Bloom filters. The protocol requires communication rounds quadratic with the number of data custodians, and the protocol is susceptible to a dictionary attack of the unique identifiers that have all the array positions in the same segment of the Bloom filter.

In Many et al.'s [29] protocol, each data custodian uses secret sharing schemes[3] [30] to split each counter position of the data custodian's Bloom filter and then distributes them to three semi-trusted third parties. The third parties use secure multiplication and comparison protocols to intersect the data custodians' Bloom filters, which adds overhead to the protocol.

Dong et al. [31] proposed a two-party protocol for private set intersection. The protocol introduced a new variant of a Bloom filter, called a garbled Bloom filter, using a secret sharing scheme. The first data custodian encodes the unique identifiers of the data custodian's records as a Bloom filter, whereas the second data custodian encodes the unique identifiers of its records as a garbled Bloom filter. Then, the data custodians intersect their Bloom filters using an oblivious transfer technique (OT)[4] [32], which adds significant overhead to the overall performance of the protocol.

Karapiperis et al. [33] proposed multi-party protocols for a secure intersection based on the Count-Min sketch.[5] Each data custodian locally encodes the unique identifiers of its records based on the Count-Min sketch, denoted as the local synopsis, and then, the data custodians jointly compute the intersections of the local synopses using a secure sum protocol. The authors proposed two protocols that use secure sum protocols based on additive homomorphic encryption [34] and obscure the secret value with a random number [19, 35]. The protocols protect only the data custodians' privacy, whereas our protocol protects individuals' and data custodians' privacy. The additive homomorphic encryption adds computation and communication overhead as the number of records and data custodians increases.

The results of the protocols in [28, 29, 31, 33] contain the probability of a false positive. Although the protocols can choose a small false positive probability, for some applications, a false positive probability may not be acceptable.

Use case

The need for comprehensive and timely infectious disease surveillance is fundamental for public health monitoring that makes early prevention and control of disease outbreaks possible. EHRs have been used as a data source for routine syndromic and laboratory-based public health surveillance [5–7].

The use case considered in this paper is distributed disease surveillance [6]. In particular, we consider the Snow system that is used for experimental evaluations of the protocol proposed in this paper [36]. The Snow system uses microbiology laboratory test results from multiple microbiology laboratories in Norway. The laboratories collect and analyze samples from patients in primary care settings, such as general practitioner offices and nursing homes.

Every day, the Snow system extracts anonymized test results and maintains the datasets within local databases at each laboratory according to a predefined data model. The data extracts contain attributes, such as infectious agent, age, sex, geographic area, and time. The Snow system broadcasts a query across the laboratories and reveals only the number of matching patients at each laboratory. We extend the Snow system with a secure sum protocol to hide the local count of a single laboratory [20].

Consider the statistical query of the count of positive or negative test results for a disease in a particular stratum of individuals (e.g., male or female) within a VD. A simple summation of the laboratories' local counts gives an overestimated count when the test results are duplicated across the laboratories. A laboratory may transfer test samples to another laboratory when the first laboratory does not have the appropriate laboratory equipment. Then, when the test results are sent to the first laboratory, the same test result appears in both laboratories' datasets.

In the context of infectious disease surveillance, two or more separate tests for an individual that have a positive result can also be considered duplicates depending on the required aggregate query for the dataset, such as the number of patients who have had a particular disease and the number of disease episodes.

Individuals may be infected with the same disease multiple times within a given time period, which may lead to being tested for the same disease at multiple laboratories. Individuals can also be tested at multiple laboratories for the same infection; this practice is more common in chronic infections. Testing at multiple laboratories may occur when patients switch healthcare providers, receive emergency care, or visit different providers during an episode of infection [37]. In Norway, primary care institutions may send samples collected from a patient to different laboratories, and patients can change general practitioners up to twice a year.

Consider a statistical query of the number of individuals infected with influenza A viruses within the VD shown in Fig. 1. The query requires that patient P_1 is counted once, even if the patient has two positive test results at data custodians D_1 and D_3. For this query, the objective of the deduplication is to link the positive test results for each individual in the VD and to maintain the test result at only one of the laboratories.

When the number of disease episodes is counted, the number of positive test results for different disease episodes for an individual across the laboratories is counted separately. However, the positive test results for an individual in the same disease episode should be counted once. For example, Lazarus et al. [38] grouped two healthcare service encounters for a patient for a lower respiratory infection into one episode if the subsequent visit occurred within six weeks of the preceding visit. The researchers assumed that the second visit likely represented a follow-up visit for the same infection. In this context, the objective of deduplication is to link an individual's positive test results for the same disease episode and keep the test result at only one of the laboratories.

We describe the protocol proposed in this paper in the context of deduplicating a VD to be able to accurately compute the statistical count of the number of individuals infected with the disease in question. However, the protocol can be easily extended to other types of statistical count queries.

Problem statement and definitions

In this section, we define the context for the deduplication problem and the problem statement.

Data custodian (D_i)

We assume three or more data custodians (e.g., hospitals, general practitioner offices, or medical laboratories) are willing to share their data for a secondary use in a health study but are concerned about privacy risks. The data custodians form a distributed health research network denoted by $\mathcal{D} = \{D_1, , D_2, \ldots, , D_N\}$, where D_i is a data custodian.

Data schema

The heterogeneity of data models is a challenge in reusing data from multiple data custodians [39]. Therefore, the distributed data must be harmonized through standardization. For example, several distributed health research networks, such as Mini-Sentinel [40] and the Shared Health Research Information Network (SHRINE) [41], create a common data model by transforming the data at each data custodian into a predefined common data model and data representations [9].

In this paper, for simplicity, we assume a common data model exists across the data custodians that enforces uniform attribute naming conventions, definitions, and data storage formats. We also assume the data distributed across the data custodians are horizontally partitioned in that each data custodian D_i collects the same attributes for a set of patients.

Virtual dataset (VD)

We assume the data query for a particular study can be broadcast to all data custodians $\mathcal{D}$. Then, each data custodian executes the query and stores a copy of the query result locally. The data extracts across the data custodians form a virtual dataset. We make the same assumption as above that the VD adheres to a common data model.

Record linkage

We consider deterministic record linkage algorithms in which a set of records belongs to the same person if they exactly or partially match on a predefined combination of identifiers. First, we describe the protocol proposed in this paper by assuming the existence of a common unique identifier j for a patient denoted by p_j. Second, we extend the protocol for deterministic record linkage using quasi-identifiers, when the available unique identifier is low quality or does not exist.

Problem statement

Assume a subset of data custodians $\mathcal{D}_s \subseteq \mathcal{D}$. Each data custodian $D_i \in \mathcal{D}_s$ has a record r_j of patient p_j in a virtual dataset. The problem addressed in this paper is to find a privacy-preserving protocol through which the patient's duplicate records are identified and removed from the virtual dataset while one occurrence of the record is maintained at one of the data custodians.

Methods

Overview

Figure 2 shows an overview of the methods we used to develop and evaluate the secure deduplication protocol

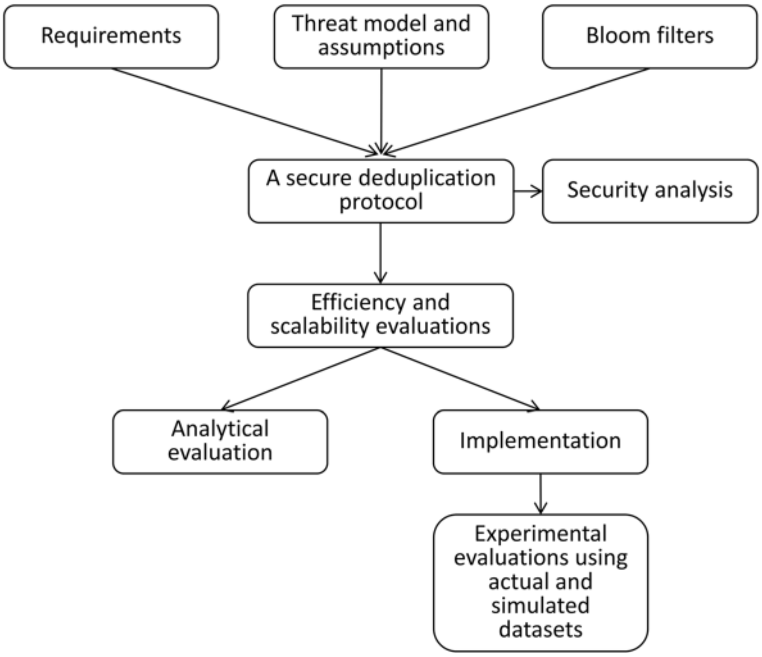

Requirements
Threat model and assumptions
Bloom filters
A secure deduplication protocol
Security analysis
Efficiency and scalability evaluations
Analytical evaluation
Implementation
Experimental evaluations using actual and simulated datasets

Fig. 2 An overview of the methods for developing and evaluating the proposed protocol

proposed in this paper. First, we defined the requirements for the protocol and the threat model and assumptions with which the protocol would be secure. We presented the building blocks used in the protocol, such as a Bloom filter, functions for the basic operations of Bloom filters, and secure sum protocol, and described the proposed protocol.

We then performed a security analysis of the proposed protocol. We also conducted theoretical and experimental evaluations of the protocol's efficiency and scalability. We implemented a prototype of the protocol and ran the experiments on the virtual datasets distributed across three Norwegian microbiology laboratories. We also ran experiments on simulated datasets with up to 20 data custodians and one million records.

Requirements for secure deduplication protocol

Data custodians' privacy concerns about disclosing patient data continue, even in the context of a pandemic [42]. Therefore, a deduplication protocol should protect the privacy of patients who have records in a VD.

However, even when patients' privacy is protected, data custodians (e.g., clinicians and health institutions) have expressed concerns about their own privacy risks [7]. For example, deduplication may reveal the total number of patients in a data custodian who satisfy certain criteria. Although this information does not directly reveal any information about the patients, data custodians might consider this information sensitive, and in many scenarios, it needs to be hidden.

For example, a general practitioner may fear that the number of laboratory test requests and results she sent to laboratories could be used to evaluate her testing behavior. A microbiology laboratory may fear that other laboratories and investors may use the number of tests the laboratory performs during a time period to gain competitive advantage. Therefore, the protocol should be designed in such a way that the total number of patients remains private.

The protocol should allow only a data custodian to learn which of its records have duplicates in the VD, which does not reveal any new sensitive information to the data custodian. However, the identity of the data custodians that contributed the duplicate records should remain unknown.

For example, in Fig. 1, the influenza A–positive test results for patient P_1 are stored at D_1 and D_3. Data custodian D_2 cannot learn any information about P_1. D_1 and D_3 learn only that P_1 tested positive for influenza A at another anonymous laboratory, which is not sensitive information.

Often, public health studies require a large number of patients' data from several data custodians. Therefore, the deduplication protocol should be computationally efficient and scale with the number of records and participating data custodians.

Threat model and assumptions

We considered a semi-honest (honest-but-curious) adversarial model in which the data custodians correctly follow the protocol specifications using the data custodians' correct data. However, the data custodians might use the messages exchanged during the protocol execution to learn information that otherwise should remain private. The adversarial model allows efficient and scalable protocols, whereas the malicious adversarial model provides stronger security at the expense of significant computation and communication costs [43–46].

We also assumed that a semi-trusted third party (STTP), denoted as the *coordinator*, who participates in the protocol without any input. In addition, we assumed that the *coordinator* follows the protocol specification and does not collude with a data custodian. An efficient and scalable protocol can be constructed using an STTP [7, 47].

We assumed that the communications between two entities that participate in the protocol are secure. Therefore, an adversary cannot read the messages sent between two honest entities, and the integrity of the messages is verified.

Bloom filter

A Bloom filter (*BF*) is a space-efficient probabilistic data structure that encodes a set $\mathcal{X}$ of n elements [48]. A *BF* is an array of size m, and each array position has one bit initially set to 0. The Bloom filter allows insertion and membership queries of an element $x \in \mathcal{X}$.

Bloom filter operations are performed using k independent hash functions $H_h(.)$, where $1 \leq h \leq k$. First, the hash of an element x is computed using each hash function $H_h(x)$. Second, the modulo m of each hash value is computed to get k array positions of *BF*, $b_h(x) = H_h(x)\ mod\ m$, where $b_h(x) \in [0, m-1]$. Then, x is inserted into the *BF* by setting all the positions $b_h(x)$ of *BF* to 1. The element x is concluded to be a non-member of the *BF* if at least one of the positions $b_h(x)$ of the *BF* is 0.

A membership query result can be a false positive due to the hash collisions that occur when all the positions $b_h(x)$ of the *BF* have been set to 1 as a result of the insertion of other elements. After elements equal to the expected number of elements n are inserted into the *BF* the false positive probability of a membership query is equal to $P(false\ positve) \approx (1 - e^{-kn/m})^k$ [49]. Figure 3 presents an example of a Bloom filter through inserting and querying elements.

A counting Bloom filter (*CBF*) is an extension of a Bloom filter [49, 50] that supports the deletion of elements, as well as insertion and membership queries. Each array position of a CBF has a counter size c greater than one bit that is large enough to avoid counter overflow.

An element x is inserted into the *CBF* by incrementing all the counters at the array positions $b_h(x)$ of the *CBF* by 1. Similarly, an element x is deleted from the *CBF* by decrementing all the counters at positions $b_h(x)$ by 1.

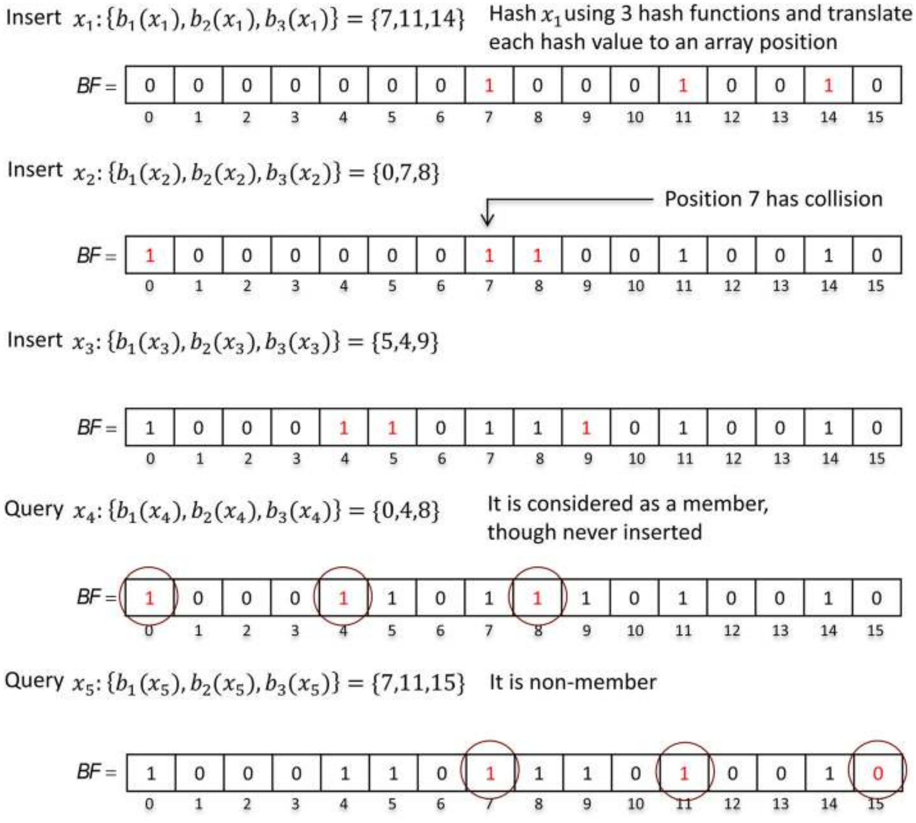

Insert $x_1: \{b_1(x_1), b_2(x_1), b_3(x_1)\} = \{7,11,14\}$
Hash x_1 using 3 hash functions and translate each hash value to an array position
BF = 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Insert $x_2: \{b_1(x_2), b_2(x_2), b_3(x_2)\} = \{0,7,8\}$
Position 7 has collision
BF = 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Insert $x_3: \{b_1(x_3), b_2(x_3), b_3(x_3)\} = \{5,4,9\}$
BF = 1 0 0 0 1 1 0 1 1 1 0 1 0 0 1 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Query $x_4: \{b_1(x_4), b_2(x_4), b_3(x_4)\} = \{0,4,8\}$
It is considered as a member, though never inserted
BF = 1 0 0 0 1 1 0 1 1 1 0 1 0 0 1 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Query $x_5: \{b_1(x_5), b_2(x_5), b_3(x_5)\} = \{7,11,15\}$
It is non-member
BF = 1 0 0 0 1 1 0 1 1 1 0 1 0 0 1 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Fig. 3 Insertion and membership query for a Bloom filter (BF) ($m = 16$, $k = 3$). $b_i(x_j)$ denotes an array position for x_j with a hash function denoted as $H_i(.)$

The element x is concluded to be a non-member of the *CBF* if at least one of the positions $b_h(x)$ is 0. A membership query has the same false positive probability as a Bloom filter.

Notations

In this section, we describe the notations that are used in the remainder of the paper. I_i denotes a set of the unique identifiers (IDs) of the records of a data custodian $D_i \in \mathcal{D}$ in a particular deduplication query. The union of all the IDs of the records across all data custodians is denoted as $S = I_1 \cup I_2 \cup \ldots \cup I_N$. We write $S \cap I_i$ and $S \cup I_i$ to denote the intersection and the union between sets S and I_i, respectively.

We use CBF_s to denote a counting Bloom filter that encodes set S and use CBF_I^i and BF_I^i to denote a counting Bloom filter and a Bloom filter that encode set I_i, respectively. $CBF_{S \cap I}^i$ and $CBF_{S \cup I}^i$ encode sets $S \cap I_i$ and $S \cup I_i$, respectively.

We use CBF_r^i to denote the random counting Bloom filter of data custodian D_i and use CBF_R to denote the sum of all the random counting Bloom filters, $\sum_{i=1}^{N} CBF_r^i$. However, CBF_R^i denotes the partial sum of the random counting Bloom filters, $\sum_{j=1}^{i} CBF_r^j + CBF_r^0$, where CBF_r^0 denotes the initial random counting Bloom filter of the leader data custodian D_L.

The union of CBF_I^i and CBF_r^i is denoted as $CBF_{r \cup I}^i$, and the Bloom filter representation of $CBF_{r \cup I}^i$ is denoted as $BF_{r \cup I}^i$. $CBF_{R \cup S}$ denotes the union of CBF_R and CBF_S.

Set operations on Bloom filters

Table 1 describes the main functions for the set operations on Bloom filters that are required for the construction of our protocol. Interested readers are referred to Additional file 1 for detailed descriptions and the algorithms of the functions.

Secure sum protocol

Several secure sum protocols are constructed using different building blocks [7, 17, 51, 52]. Secure sum protocols compute $s = \sum_{i=1}^{N} v_i$, where $v_i \in [0, m)$ is the secret value of data custodian D_i. The protocols compute without disclosing v_i to any entity that participates in the protocol. We extend the secure sum protocol proposed in [19, 35] to compute the union of random counting Bloom filters, $CBF_R = \sum_{i=1}^{N} CBF_r^i$, where CBF_r^i is the random counting Bloom filter of D_i. Assume that D_1 is selected as the leader data custodian, denoted as D_L. The protocol steps are shown in Algorithm 1.

In steps 1–3, the leader data custodian D_L computes $CBF_R^1 = add(CBF_r^1, CBF_r^0)$ and sends the result CBF_R^1 to data custodian D_2. In steps 4–11, each data custodian D_i, in turn, computes $CBF_R^i = add(CBF_r^i, CBF_R^{i-1})$ where $2 \le i \le N$ and CBF_R^{i-1} is the value received from the previous data custodian D_{i-1}. Then, D_N sends its result CBF_R^N to D_L. In step 12, D_L computes $CBF_R = sub(CBF_r^N, CBF_r^0)$ and gets the actual sum $CBF_R = \sum_{i=1}^{N} CBF_r^i$. In steps 13–15, D_L broadcasts CBF_R to all data custodians.

In this protocol, collusion between data custodians D_{i-1} and D_{i+1} reveals the secret value of D_i. Extensions to the protocol are proposed in [21, 53] to make collusion between data custodians difficult.

A secure deduplication protocol

In this section, we describe the secure deduplication protocol proposed in this paper. The protocol includes the setup and computation phases.

Setup phase

In this phase, the *coordinator* broadcasts a start message that contains the user query criteria and the *P*(*false positive*) value to each D_i in $\mathcal{D}$. Then, the data custodians jointly select the leader data custodian, denoted as D_L. For simplicity, let us assume that D_1 is selected as the leader. Then, they form a ring topology, $D_L \rightarrow D_2 \rightarrow D_3 \rightarrow \ldots \rightarrow D_i \rightarrow D_{i+1} \rightarrow \ldots \rightarrow D_N$, as shown in Fig. 4.

The data custodians jointly select the required parameters, such as the expected number of records n, Bloom filter size m, number of hash functions k, counter size c, and *P*(*false positive*). The data custodians also agree on a cryptographic hash function $H_0(.)$ and the k hash functions H_k with two secret keys k_0 and k_1.

Computation phase

The computation phase contains two subprotocols, such as the secure duplicate identifier and the distributed

Table 1 Functions for basic operations of Bloom filters

Functions	Description
add(CBF_r^i, CBF_I^i)	Returns counting Bloom filter $CBF_{r \cup I}^i$ that represents the summation of CBF_r^i and CBF_I^i
sub($CBF_{R \cup S}$, CBF_R)	Returns counting Bloom filter CBF_S that represents the subtraction of CBF_R from $CBF_{R \cup S}$
intersect(CBF_S, BF_I^i)	Returns counting Bloom filter $CBF_{S \cap I}^i$ that represents the intersection between CBF_S and BF_I^i
count($CBF_{S \cap I}^i$, $\{b_1(x), b_2(x), \ldots, b_k(x)\}$)	Returns f that is equal to the number of occurrences of x in $CBF_{S \cap I}^i$
toBloomFilter(CBF_I^i)	Returns Bloom filter BF_I^i that represents CBF_I^i

```
Algorithm 1: secureSum({$CBF_r^0, CBF_r^1, CBF_r^2, \ldots, CBF_r^N$})

Input: random counting Bloom filters $CBF_r^i$ of each $D_i$ including $D_L$, the random counting Bloom filter $CBF_r^0$ of $D_L$, and ring topology $\mathcal{D}$
Output: a counting Bloom filter $CBF_R = \sum_{i=1}^{N} CBF_r^i$ that is the union of the random counting filters of all data custodians

$D_L: CBF_R$                                      //Note: $D_L$ and $D_1$ are the same. $D_L$ creates a counting Bloom filter
$D_L: CBF_R^1 = add(CBF_r^1, CBF_r^0)$           //$D_L$ adds its two random filters.
$D_L: CBF_R^1 \rightarrow D_2$                    //$D_L$ sends $CBF_R^1$ to $D_2$.
for $i = 2$ to $N$ do
$D_i: CBF_R^i = add\ (CBF_R^{i-1}, CBF_r^i)$      //$D_i$ adds $CBF_r^i$ and $CBF_R^{i-1}$
if ($i < N$) then
        $D_i: CBF_R^i \rightarrow D_{i+1}$        //$D_i$ sends $CBF_R^i$ to $D_{i+1}$
else if ($i = N$) then
        $D_i: CBF_R^N \rightarrow D_L$            //$D_N$ sends $CBF_R^N$ to $D_L$
end
end
$D_L: CBF_R = sub(CBF_r^N, CBF_r^0)$             //$D_L$ subtracts $CBF_r^0$ from $CBF_R^N$
for $i = 2$ in $N$ do
        $D_L: CBF_R \rightarrow D_i$              //$D_L$ broadcasts $CBF_R$ to each $D_i$
end
```

sorted neighborhood. The secure duplicate identifier subprotocol allows each data custodian to learn which of its records have duplicate records in the VD with false positive probability *P*(*false positive*). Then, the distributed sorted neighborhood subprotocol is executed on the results of the secure duplicate identifier subprotocol to identify the real duplicate records and remove the duplicate records while maintaining a single occurrence of the records.

A secure duplicate identifier subprotocol

The objective of this subprotocol is to allow each data custodian D_i to identify which of its records has a duplicate in the VD with a small false positive probability *P*(*false positive*). The protocol consists of the following steps:

1. Each D_i in $\mathcal{D}$ performs the following steps:
   a. Extract from its local dataset a set of unique IDs, denoted as I_i, of the patients who satisfy the user query criteria.
   b. Encode I_i as the counting Bloom filter CBF_r^i using the keyed hash functions H_k with the secret key k_1.
   c. Create the random counting Bloom filter CBF_r^i (the algorithm used to create the random counting Bloom filter is described in Additional file 1).
2. D_L creates the initial random counting Bloom filter CBF_r^0.
3. The data custodians $\mathcal{D}$ jointly run Algorithm 1 to compute the sum $CBF_R = \sum_{i=1}^{N} CBF_r^i$.
4. Each D_i sums CBF_r^i and CBF_I^i and sends the result $CBF_{r \cup I}^i$ to the *coordinator.*

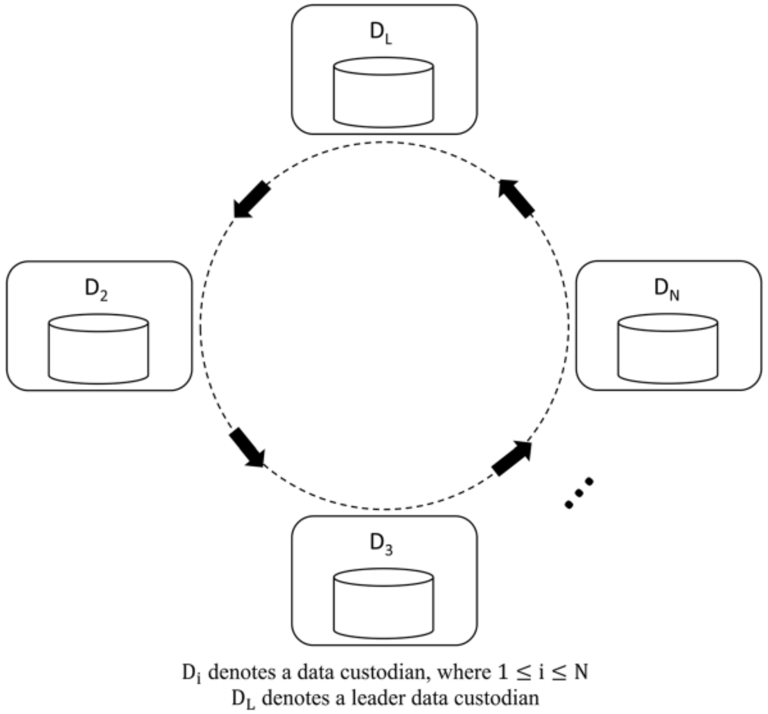

D_L
D_2
D_N
D_3
D_i denotes a data custodian, where 1 ≤ i ≤ N
D_L denotes a leader data custodian

Fig. 4 Ring topology of the data custodians

5. The *coordinator* computes the sum $CBF_{R∪S} = \sum_{i=1}^{N} CBF^i_{r\,\cup\,I}$.
6. For each D_i in $\mathcal{D}$, the *coordinator* performs the following steps:
   a. Convert $CBF^i_{r\,\cup\,I}$ into the Bloom filter $BF^i_{r\,\cup\,I}$.
   b. Intersect $CBF_{R\,\cup\,S}$ and $BF^i_{r\,\cup\,I}$ and send the result $CBF_{R\,\cup\,S} \cap BF^i_{r\,\cup\,I}$ to D_i.
7. Each D_i in $\mathcal{D}$ performs the following steps:
   a. Create the Bloom filters BF^i_I and $BF^i_{r\,\cup\,I}$ from the counting Bloom filters CBF^i_I and $CBF^i_{r\,\cup\,I}$, respectively.
   b. Intersect CBF_R and $BF^i_{r\,\cup\,I}$ and create the counting Bloom filter $CBF_R \cap BF^i_{r\,\cup\,I}$.
   c. Subtract $CBF_R \cap BF^i_{r\,\cup\,I}$ from $CBF_{R\,\cup\,S} \cap BF^i_{r\,\cup\,I}$. The result is denoted by $(CBF_{R\,\cup\,S} \cap BF^i_{r\,\cup\,I})$ – $(CBF_R \cap BF^i_{r\,\cup\,I}) = (CBF_{R\,\cup\,S} - CBF_R) \cap BF^i_{r\,\cup\,I}$. However, we know that $CBF_{R\,\cup\,S} - CBF_R = CBF_S$. Therefore, the result can be expressed by $CBF_S \cap BF^i_{r\,\cup\,I}$.
   d. Intersect $CBF_S \cap BF^i_{r\,\cup\,I}$ and BF^i_I and create the counting Bloom filter denoted as $CBF_S \cap BF^i_{r\,\cup\,I} \cap BF^i_I$. The expression can be reduced to $CBF^i_{S\,\cap\,I} = CBF_S \cap BF^i_I$, as $BF^i_{r\,\cup\,I} \cap BF^i_I$ is equal to BF^i_I.
   e. Query the number of occurrences of the IDs I_i in $CBF^i_{S\,\cap\,I}$ using the *count*() function, and create the list L_i that contains the IDs that have more than one occurrence.

In steps 1–2, each data custodian D_i (where $1 \le i \le N$) encodes the unique IDs of its records as the counting Bloom filter CBF^i_I and creates the random counting Bloom filter CBF^i_r. The leader data custodian D_L creates the additional random counting Bloom filter CBF^0_r. In step 3, the data custodians jointly compute the sum of their random counting Bloom filters, $CBF_R = \sum_{i=1}^{N} CBF^i_r$, using Algorithm 1. In step 4, D_i computes $CBF^i_{r\,\cup\,I} = add(CBF^i_r, CBF^i_I)$ and sends the result $CBF^i_{r\,\cup\,I}$ to the *coordinator*. In step 5, the *coordinator* sums all data custodians' $CBF^i_{r\,\cup\,I}$, $CBF_{R∪S} = \sum_{i=1}^{N} CBF^i_{r\,\cup\,I}$.

In step 6, the *coordinator* computes $BF^i_{r\,\cup\,I} = toBloomFilter(CBF^i_{r\,\cup\,I})$ and $CBF_{R\,\cup\,S} \cap BF^i_{r\,\cup\,I}$. The *coordinator* sends $CBF_{R\,\cup\,S} \cap BF^i_{r\,\cup\,I}$ to D_i. In step 7, each data custodian D_i creates the counting Bloom filter $CBF^i_{S\,\cap\,I} = CBF_S \cap BF^i_I$ that encodes the intersection between the IDs of D_i and all data custodians. Finally, D_i queries its IDs in $CBF^i_{S\,\cap\,I}$ to create the list L_i that contains the IDs for the records that are likely to be duplicates with the false positive probability *P*(*false positive*). Although the *P*(*false positive*) is very small, for some applications it may not be acceptable, and the true duplicate records should be identified.

The total number of likely duplicate records across the data custodians, $\sum_{i=1}^{N}|L_i|$ is very small compared to the total number of records, as the number of records that have duplicate records is often a small proportion of the total number of records. Therefore, we can run existing deterministic PPRL protocols [12, 22, 24, 27] on the results of the secure duplicate identifier subprotocol with minimal computation and communication complexity. In the next section, we present an improved protocol based on the keyed hash function that reduces the required number of comparisons.

Secure distributed sorted neighborhood subprotocol

In the conventional sorted neighborhood (SN) technique [54, 55], sorting keys are generated for each record using a single attribute or a concatenation of the attributes of the records, and the keys are sorted lexicographically. Then, a sliding window of fixed size w is moved over the sorted records, and only the records in the same window are compared.

After the secure duplicate identifier subprotocol is run, each data custodian D_i has the list L_i that contains the IDs of the likely duplicate records. Note that the size of L_i is much smaller than the total number of records of D_i. In a simple approach for finding the actual duplicate records, D_i hashes each ID in L_i using a keyed hash function and sends the result HL_i to the *coordinator*, who computes the union of the hash lists from all data custodians, $HL = \uplus_i HL_i$. Then, the coordinator performs exact matching between every ID with every ID in HL. However, in practice, we know that most of the comparisons are unlikely to match.

Let us assume that a set of data custodians $\mathcal{D}_s \subseteq \mathcal{D}$ and each data custodian $D_i \in \mathcal{D}_s$ has the record r_j with the ID j. As r_j is a duplicate record, value j appears in the list of IDs of the likely duplicate records, L_i, of each $D_i \in \mathcal{D}_s$. $H_0(j)$ denotes the hash of j with hash function $H_0(.)$. As the hash of j with the same hash function multiple times gives the same hash values, each $D_i \in \mathcal{D}_s$ sends to the *coordinator* the list HL_i that contains $H_0(j)$. Therefore, HL contains multiple occurrences of $H_0(j)$, and sorting HL brings hash values for the same ID next to each other.

Based on these observations, we present a distributed sorted neighborhood (DSN) subprotocol that extends the SN technique. The protocol parallelizes the sorting by making each data custodian D_i locally sort HL_i, and the *coordinator* merges only the sorted lists. The DSN protocol has the following steps:

1. Each D_i in $\mathcal{D}$ performs the following steps:
   a. For every ID j in L_i, D_i performs the following steps:
      1. Hash j using the keyed hash function $H_0(.)$ with the secret key k_0.
      2. Store the hash of j in the list SL_i.
   b. Lexicographically sort SL_i.
   c. Send SL_i to the *coordinator.*
2. The *coordinator* performs the following steps:
   a. Merge the SL_i of each D_i in $\mathcal{D}$ and create the list SL.
   b. Slide a window of size w over the list SL and compare each pair of hash IDs within the window. If at least two hash IDs match, then the records associated with the IDs are duplicates.
   c. Send to D_i the list DL_i of the hash IDs of the records that D_i needs to remove from its local dataset.
3. Each D_i in $\mathcal{D}$, for every ID j in DL_i, removes its record associated with j.

Extension of the secure deduplication protocol for deterministic algorithms

Thus far, the proposed protocol has been described for situations in which a common unique identifier exists, which enables efficient and high-quality linkage. This assumption is realistic in countries, such as Norway, Sweden, and Denmark, where a high-quality unique personal identifier is available [56, 57].

However, in many situations, the available unique identifier is low quality or does not exist. We describe how our protocol can be extended to support deterministic record linkage algorithms that define the criteria about which identifiers need to match in order to accept the linkage of a pair of records.

To increase the quality of the linkage, data cleaning often precedes record linkage. We also assume appropriate data cleaning occurs before the protocol is run. Various data-cleaning techniques, such phonetic encoding algorithms, have been proposed in the literature [58].

It has been shown that a linkage key can be created based on a concatenation of quasi-identifiers, such as name, sex, date of birth, and address. Studies have estimated that up to 87% of the U.S. population [59], 98% of the Canadian population [60], and 99% of the Dutch population [61] are unique, with a combination of quasi-identifiers, such as postal code, sex, and date of birth.

The National Cancer Institute in the United States uses a deterministic record linkage algorithm to link Surveillance, Epidemiology and End Results (SEER) data collected from cancer registries and Medicare claims data. The algorithm creates linkage keys using a set of criteria based on a Social Security number (SSN), first name, last name, date of birth, and sex [62, 63].

Let us consider a deterministic record linkage algorithm that has p linkage keys where each linkage is generated using a distinct match criterion defined by combinations of quasi-identifiers. For each match criterion, each data custodian creates a linkage key, and the deduplication protocol is run with the linkage key the same way the protocol is run with a unique identifier. However, in the distributed sorted neighborhood subprotocol, each data custodian sends the hash of the local identifiers of the likely duplicate records with the hash of the linkage keys to the *coordinator*. Finally, the *coordinator* identifies the actual duplicate records from the results of the protocol with all the linkage keys.

Let us consider, for simplicity of description, that each data custodian has an equal number of records. The computation time for a data custodian to create linkage keys for its records based on a combination of quasi-identifiers is denoted as t_l. The runtime for the protocol using a unique identifier is denoted as t_u. Assuming that the data custodians generate linkage keys for their records in parallel, deduplication using a linkage key has a total runtime of $t_u + t_d$.

For a deterministic record linkage algorithm that has p linkage keys, the total runtime is $p \times (t_u + t_d) + t_a$, where t_a is the sum of the additional time required to send local unique identifiers to the *coordinator* and the computation time for the *coordinator* to find the actual duplicate by combining the results of the protocol with each linkage key. However, as a separate instance of the protocol can run with each linkage key in parallel, the runtime reduces to $(t_u + t_d) + t_a$.

Results

In this section, we describe the security analysis and the implementation of the proposed deduplication protocol. We also describe the theoretical and experimental evaluations of the protocol's efficiency and scalability.

Security analysis

We prove the security of the proposed protocol in the presence of corrupt data custodians or a corrupt *coordinator* who tries to learn information as a result of the protocol execution. We assume that a corrupt *coordinator* does not collude with a corrupt data custodian.

For the security proof of the protocol, we follow the standard security definition that is called privacy by simulation. For an adversary that controls a set of data custodians (or the *coordinator*), the adversary's view (the information learned during the protocol execution) is a combination of the corrupt data custodians' views. The adversary also accesses the corrupt data custodians' inputs and outputs. Thus, in the simulation paradigm, we require to show the existence of an efficient simulator that can generate the adversary's view in the protocol execution given only the corrupt data custodians' inputs and outputs.

Theorem 1 (compromised $\mathbf{D_i}$) *The protocol is secure against an honest-but-curious adversary ADV that corrupts (or controls) at most $N - 2$ data custodians.*

Proof We prove the robustness of the protocol by looking at the exchanged messages and reducing the security to the properties of the Bloom filter.

We denote the corrupt data custodians as $\mathcal{D}_A \subset \mathcal{D}$, where $|\mathcal{D}_A| \leq N - 2$. The inputs to a simulator are the inputs and outputs of the corrupt data custodians $\mathcal{D}_A$. The inputs and outputs of each corrupt data custodian $D_a \in \mathcal{D}_A$ are the list of the IDs of its records I_a and the list of the IDs for likely duplicate records L_a, respectively.

The view of each corrupt data custodian $D_a \in \mathcal{D}_A$ are the counting Bloom filters, such as $CBF_I^a, CBF_{S\cap I}^a, CBF_r^a$ and CBF_R. CBF_I^a and $CBF_{S\cap I}^a$ can be generated from lists I_a and L_a, respectively. CBF_r^a and CBF_R are randomly generated. In general, the simulator can generate the adversary's view in the protocol execution from the corrupt data custodians' inputs and outputs. Thus, the protocol is secure against an honest-but-curious adversary so that the protocol computes without revealing anything except the outputs. Therefore, the adversary cannot extract any private information about patients who have records at honest data custodians. In addition, the adversary cannot learn the number of records at honest data custodians.

Let us assume that the ID j for a duplicate record r_j is in the list L_a of corrupt data custodian $D_a \in \mathcal{D}_A$. The adversary learns the number of duplicates for r_j from $CBF_{S\cap I}^a$ with a false-positive probability equal to $P(\textit{false positive})$, denoted as d. The adversary can look into its inputs to learn the actual number of duplicates of r_j at $\mathcal{D}_A$, denoted as d_A. Therefore, an adversary may infer whether duplicate records for r_j exist at honest data custodians with the following probability:

$$p = {}^{(d - d_A)(1 - P(\textit{false positive}))} \Big/ {}_{(N - |\mathcal{D}_A|)}.$$

Theorem 2 (compromised coordinator) *An honest-but-curious adversary ADV that corrupts the coordinator cannot infer any information about the presence or absence of patients at data custodians and the number of records contributed by a data custodian.*

Proof We prove the security of the protocol by analyzing the messages received by the *coordinator* during the execution of the protocol and reduce its security to the properties of the hash functions H_k and $H_0(.)$ used in the protocol.

The *coordinator's* view is the counting Bloom filter $CBF_{r\cup I}^i$ and the list of the hash IDs of the likely duplicate records SL_i of each data custodian D_i. The *coordinator* does not have inputs and outputs. The objective of the security proof is to show that private information about patients and data custodians cannot be learned based on the *coordinator's* view during the protocol execution.

The secret keys (k_0, k_1) used by the data custodians are not available to the simulator. Therefore, the simulator cannot learn the IDs of the records inserted in $CBF^i_{r \cup I}$. In addition, as the hash function $H_0(.)$ is cryptographically secure, the simulator cannot learn the IDs based on SL_i.

Each array position of $CBF^i_{r \cup I}$ has a counter value equal to the sum of the corresponding counter values of CBF^i_I and CBF^i_r. Thus, the counter values of $CBF^i_{r \cup I}$ are random, as every counter position of CBF^i_r has a random value. The random noise CBF^i_r inserted in $CBF^i_{r \cup I}$ prevents the simulator from learning the approximate total number of records of D_i encoded by CBF^i_I.

Therefore, *ADV* cannot learn the IDs and the number of records held at a data custodian, and consequently, the protocol is secure in the face of a corrupt *coordinator*.

Implementation

A prototype of the proposed deduplication protocol is implemented in Java. The prototype contains the local and *coordinator* software components. The local software component is deployed at each data custodian, while the *coordinator* software component is deployed at the *coordinator*. The parameters required for an instance of the protocol are configured through the configuration files.

The dataset at each data custodian was stored in a MySQL relational database. We used the JavaScript Object Notation (JSON) format for message communication. Each component used an Extensible Messaging and Presence Protocol (XMPP) [64] client to connect to an XMPP server. Then, a JSON message was sent through the XMPP server between two entities that participate in the protocol. All messages were compressed using the Lz4 [65] lossless compression algorithm to reduce the overall size. After transmission, each message was decompressed before actual use.

Analytical evaluation

The main concerns for the practical use of SMC protocols are efficiency and scalability. Efficiency is the ability to compute with a good performance, which is often expressed by the communication and computation complexity. Communication complexity is analyzed in terms of the communication rounds and the size of the messages exchanged between the parties involved in the protocol. For N data custodians, each data custodian sends three messages and receives three messages, except the leader data custodian that sends $N + 2$ messages. The *coordinator* sends $2N$ messages and receives $2N$ messages. The overall communications of the protocol is $6N - 1$, which is linear with the number of data custodians $O(N)$.

The size of a message that contains a counting Bloom filter depends on the Bloom filter size m and counter size c. The size of the message that contains the list of likely duplicate hash IDs SL_i is small compared to the size of the message that contains the IDs I_i of all records, but it depends on the false positive probability and the proportion of the records of D_i that have duplicates in the virtual dataset.

Computation complexity is measured in terms of the time it takes for each entity to complete local computations and the protocol runtime. Scalability is measured in terms of the change in efficiency as the number of records and data custodians increases.

In general, the local computations of the protocol are computationally very efficient, as it does not use a building block that adds overhead to the performance. Bloom filter operations require only $O(1)$ time. The other computations are performed only on the list of the hash IDs of the likely duplicate records, which are a small proportion of the IDs of all the records. In addition, the data custodians often compute in parallel. Detailed analysis of the total computation time is provided in Additional file 1.

Experimental evaluation

We ran the experiments using actual and simulated datasets to evaluate the efficiency and scalability of the protocol. The experiments were run 100 times, and the average total runtime for the protocol and the local computation time for each entity were recorded. In this section, we report only the total runtimes. Details regarding the parameters used for the experiments are given in Additional file 1.

In situ experiments

We deployed the prototype at three microbiology laboratories located in Norway on top of the Snow disease surveillance system [36]. Two laboratories are departments at the University Hospital of North Norway (UNN) and Nordland Hospital (NLSH). The third laboratory is Fürst Medical Laboratory, a private laboratory. UNN and NLSH together cover a total population of more than 475 000 inhabitants. Based on the number of laboratory tests conducted within a specific time period, we estimated that the Fürst population coverage is approximately twice the total population covered by UNN and NLSH.

At Fürst, the local software component was deployed on an Intel i5-4590 3.3GHz quad core with 8GB RAM and Ubuntu 14.04. At UNN and NLSH, the local software component was deployed on an Intel Xeon E5-2698 v3 2.3GHz dual core with 4GB RAM and Red Hat Enterprise Linux 7. The coordinator software component was deployed on an Intel Xeon X3220 2.4GHz quad core with 8GB RAM and Ubuntu 14.04. The laboratories and the *coordinator* were connected through the

Norwegian Health Network, a wide area network of healthcare service providers. Details about the network connections and the communication patterns are described in Additional file 1.

We ran two experiments on the data distributed across the three laboratories. The first experiment involved answering a query about the number of individuals infected with influenza A between January 2015 and April 2016. Each laboratory locally queried the IDs of the individuals who were tested positive for influenza A during this time period, and a virtual dataset that contained 5329 records was created.

The second experiment involved answering a query about the number of patients who were tested at multiple laboratories between January 2015 and April 2016. Each laboratory locally queried the unique IDs of the patients who had been tested for any of the diseases included in the Snow system during the time period, and a virtual dataset that contained 85 353 unique IDs was created.

We divided each virtual dataset into segments by varying the time period in which the analyses were performed. Then, we ran the protocol on each segment of the virtual datasets.

Figures 5 and 6 show the runtimes for the protocol on the two virtual datasets as the total number of records increased. The deduplication of 5329 and 85 353 records took around 0.8 and 7.5 s, respectively. The local computation time for the laboratories and the *coordinator* is presented in Additional file 1. For the deduplication of 85 353 records, the local computation time for the *coordinator* and the data custodians did not exceed 0.6 s.

The experiment on the VD of the positive test results for influenza A found one patient who was tested at multiple laboratories between January 2015 and April 2016. The experiment on the VD that contained the test results for various diseases found 449 patients who had been tested at multiple laboratories for different infectious diseases. The results showed that the samples collected from each patient were tested at multiple laboratories at different times.

In vitro experiments

We deployed the prototype of the protocol on a cluster computer. Each node had two Intel E8500 dual-core 1.17GHz CPUs, 4GB RAM, and CentOS 6.7. The nodes were connected through fast Ethernet.

We ran experiments on simulated virtual datasets that consisted of a large number of data custodians and records. The VDs consisted of a varying number of data custodians (i.e., 5, 10, 15, and 20) and total number of records (i.e., 200 000, 400 000, 600 000, 800 000, and 1 000 000). The total number of records of each VD was distributed equally among all the data custodians, and each data custodian contained around 5% duplicate records. Details about the datasets are provided in Additional files 1 and 2.

Figures 7 and 8 show the total runtime for the protocol as the total number of records in the virtual dataset and the number of participating data custodians increased, respectively. The deduplication of one million records distributed across five and 20 data custodians took around 34 and 45 s, respectively. The local computation time for the laboratories and the *coordinator* is presented in Additional file 1. In general, the local computation time for the *coordinator* and the data custodians did not exceed 34 and 7 s, respectively.

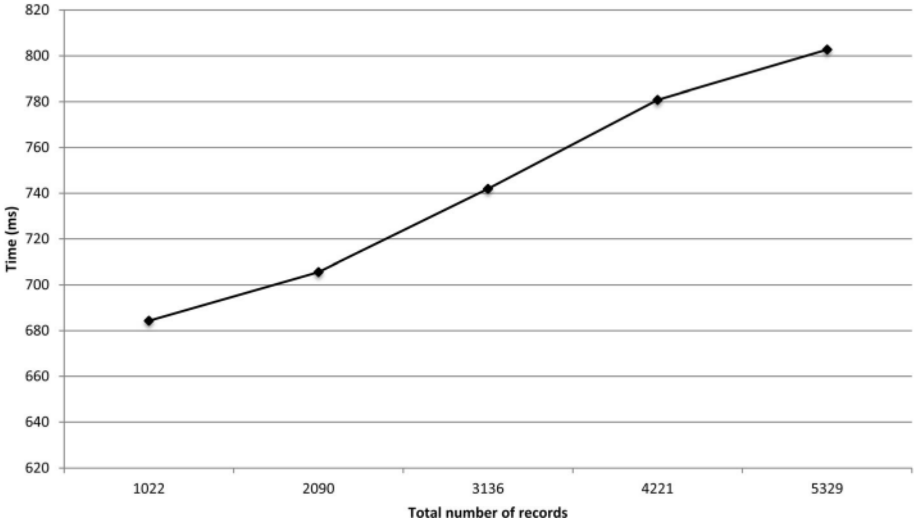


Fig. 5 The total runtime for the protocol on the influenza A datasets

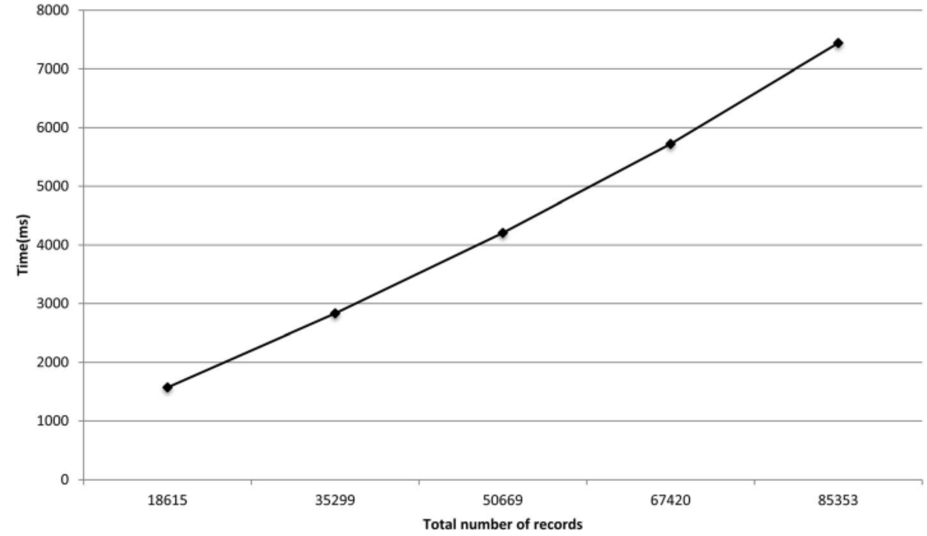


Fig. 6 The total runtime for the protocol on the datasets that contain the test results for various diseases

Discussion

We proposed a privacy-preserving protocol for the deduplication of data horizontally partitioned across multiple data custodians. The protocol protects the privacy of patients and data custodians under the semi-honest adversarial model. The protocol remains secure even when up to $N - 2$ data custodians collude.

The protocol satisfies the security requirements that were formulated in [26] for a secure record linkage protocol. However, we assumed that the *coordinator* has no means of getting the secret keys used in the protocol, which improves the efficiency and scalability of the protocol. The assumption can be ensured through a data use agreement among the data custodians that prohibits them from sharing the secret keys with the *coordinator*.

The protocol was deployed and evaluated in situ for the deduplication of test results distributed across three microbiology laboratories. The protocol was also evaluated in vitro on simulated microbiology datasets of up to one million records and 20 data custodians. The deduplication of the one million simulated records distributed across 20 data custodians was completed within 45 s. The experimental results showed that the proposed protocol is more efficient and scalable than previous protocols [29, 31].

The local computation time for the entities and the total runtime for the protocol scale linearly as the number

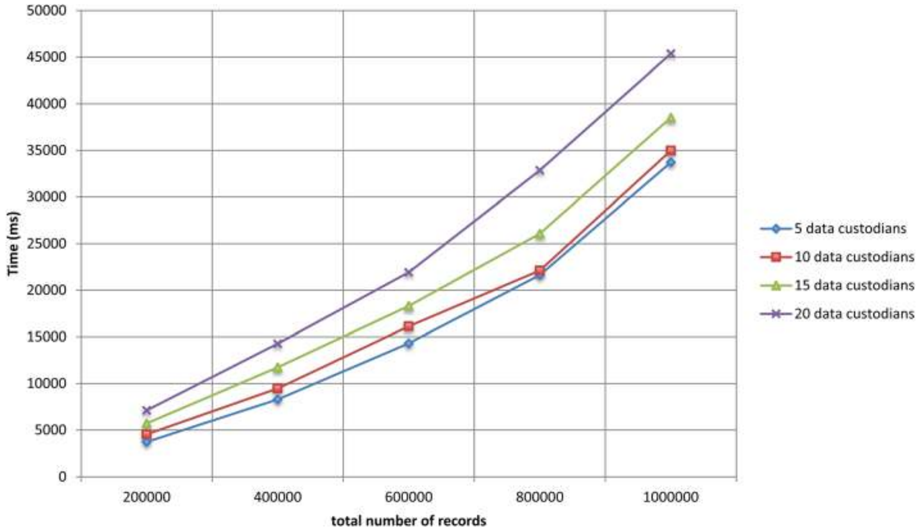


Fig. 7 The total runtime for the protocol on the simulated datasets as the total number of records increases

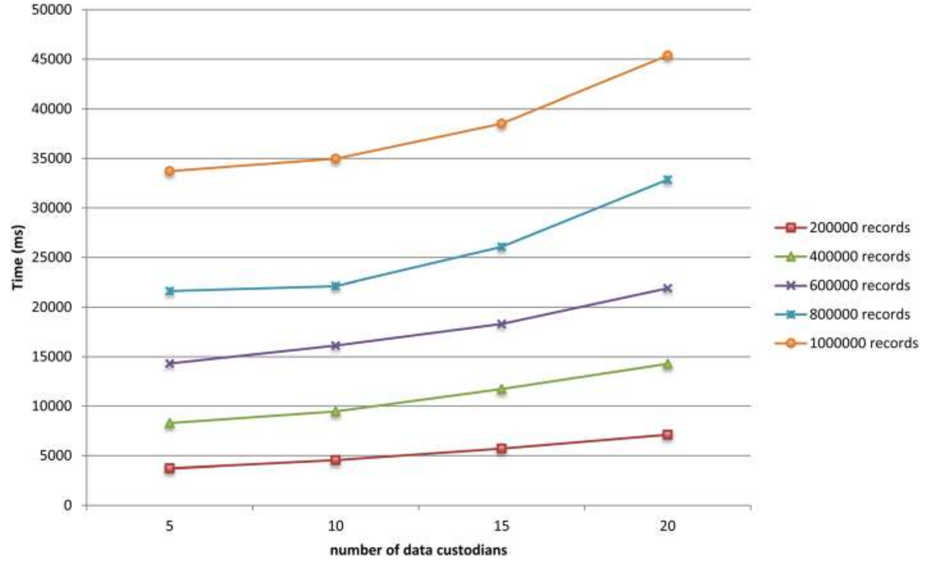


Fig. 8 The total runtime for the protocol on the simulated datasets as the number of participating data custodians increases

of participating data custodians and the total number of records increase. The protocol scales because Bloom filter operations are not expensive, and the *coordinator* and data custodians perform most of the computations in parallel. In addition, the number of communication rounds of a data custodian is constant and does not increase with the addition of new data custodians.

The protocol was not experimentally evaluated for situations in which there is no unique identifier. However, we theoretically showed that the protocol remains scalable. The computation complexity of each party linearly increases with the number of steps of the deterministic record linkage algorithm.

There is a need for reuse health data at the regional, national, and global levels to increase the number of records and participating data custodians for a given study [66, 67]. The blocking technique [68] can be applied to increase the scalability of the proposed deduplication protocol to such a large scale. The intuition for the use of the blocking technique is that running a separate instance of the protocol on a subset of records that are likely to match enables parallel computations. A very simple example is running different instances of the protocol on the records of female and male patients.

In practice, data custodians create blocking keys for their records based on one or more attributes, and records that have the same blocking key values are grouped into the same block, which consequently divides the virtual dataset into segments. Then, the data custodians jointly execute a separate instance of the protocol for each segment of the virtual dataset in parallel. The data custodians can execute the protocol instances on different CPU cores or servers to increase the scalability of the protocol.

Conclusions

Deduplication is a necessary preprocessing step for privacy-preserving distributed statistical computation of horizontally partitioned data. However, deduplication should not reveal any private information about individuals and data custodians. We proposed a privacy-preserving protocol for the deduplication of a horizontally partitioned dataset.

Efficiency and scalability are the main challenges for practical uses of SMC protocols. The experimental evaluations of the proposed protocol demonstrated its feasibility for a wide range of practical uses.

As discussed in the Discussion section, we plan to parallelize the execution of the protocol using the blocking technique. Furthermore, we also plan to integrate the protocol with the privacy-preserving distributed statistical computation framework we developed [20].

Endnotes

[1] Commutative encryption is a form of encryption in which the order of the consecutive encryption and decryption of a value with different cryptographic keys does not affect the final result and no two values have the same encrypted value [69].

[2] The protocols are different from the protocol proposed in [70, 71] for probabilistic record linkage. In Schnell et al.'s protocol [70], for each record, each identifier is encoded as a separate Bloom filter, whereas in Durham et al.'s protocol [71], to avoid frequency-based cryptanalysis, the set of identifiers of each record is encoded as a Bloom filter.

[3] Secret sharing is a method by which a secret value is split into shares and a predefined number of shares are required to reconstruct the secret value [30].

[4]OT is a method for two parties to exchange one of several values in which the sender is oblivious to which value is selected, while the receiver learns only the selected value [32].

[5]The Count-Min sketch [72] is, similar to the counting Bloom filter (see the description of the counting Bloom filter in the Methods section), a space-efficient probabilistic data structure for encoding a set of elements that allows querying the frequencies of the occurrence of the inserted elements with some error.

Additional files

Additional file 1: It contains a description of the Bloom filter set operations, the computation complexity analysis of the protocol, the algorithm for generating random Bloom filters, the parameters used in the experiments, the datasets used in the experiments, the network connection for the in situ experiments, and additional experiment results. (DOCX 6563 kb)

Additional file 2: It is a Bash script that implements the algorithm we used to generate the simulated microbiology datasets. The script is discussed in Additional file 1. (SH 9 kb)

Abbreviations

EHR: Electronic Health Record; JSON: JavaScript Object Notation; PPRL: Privacy-Preserving Record Linkage; SMC: Secure Multi-Party Computation; VD: Virtual Dataset; XMPP: Extensible Messaging and Presence Protocol

Acknowledgments

We would like to thank Gro Berntsen for discussions about the use case section. We also would like to thank Gunnar Hartvigsen and Andrius Budrionis for invaluable discussions. We must thank the system developers of the Snow system, in particular Torje Henriksen, for their invaluable support in the integration of the protocol into the Snow system and the execution of the experiments across the microbiology laboratories.
We would like to thank the microbiology laboratories of University Hospital of North Norway, Nordland Hospital, and Fürst Medical Laboratory for permitting the in situ experiments to be performed on their datasets. We are also indebted to Eyvind W. Axelsen and Haagen Berg at the microbiology laboratories for their support in the preparation of the microbiology datasets used for the in situ experiments.

Funding

This work was supported by the Center for Research-based Innovation, Tromsø Telemedicine Laboratory (TTL), through The Research Council of Norway, grant number 174934. The study was also partially supported by the Research Council of Norway, grant number 248150/O70. The funding bodies did not have any role in the design and evaluation of the protocol and in writing the manuscript.

Availability of data and material

The laboratory datasets used for the in situ experiments of this study are available from University Hospital of North Norway, Nordland Hospital, and Fürst Medical Laboratory, but restrictions apply to make these data publicly available. However, the algorithm that generated the datasets used for the in vitro experiments is included in the supplementary files of this article. The prototype software of the protocol is available from the corresponding author upon request.

Authors' contributions

KYY contributed to the conception and design of the manuscript, and was the major contributor to the writing of the manuscript. KYY also implemented the protocol, contributed to the design of the experiments, and executed the experiments. AM contributed to the design of the manuscript and participated in drafting the manuscript. JGB contributed to the conception of the manuscript and the design of the experiments. JGB also extensively reviewed the manuscript for important scientific content. All authors read and approved the submission of the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

The privacy ombudsman of the University Hospital of North Norway approved the in situ experiments, and all participating laboratories also agreed to the experiments. Individuals' consent was not required.

Author details

[1]Department of Computer Science, UiT The Arctic University of Norway, 9037 Tromsø, Norway. [2]Norwegian Centre for E-health Research, University Hospital of North Norway, 9019 Tromsø, Norway. [3]Department of Computer Science, University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK. [4]Department of Clinical Medicine, UiT The Arctic University of Norway, 9037 Tromsø, Norway.

Received: 27 July 2016 Accepted: 10 November 2016
Published online: 03 January 2017

References

1. Ross MK, Wei W, Ohno-Machado L. "Big data" and the electronic health record. IMIA Yearb. 2014;9:97–104.
2. Kohane IS, Drazen JM, Campion EW. A glimpse of the next 100 years in medicine. N Engl J Med. 2012;367:2538–9.
3. Geissbuhler A, Safran C, Buchan I, Bellazzi R, Labkoff S, Eilenberg K, et al. Trustworthy reuse of health data: a transnational perspective. Int J Med Inf. 2013;82:1–9.
4. Hripcsak G, Bloomrosen M, FlatelyBrennan P, Chute CG, Cimino J, Detmer DE, et al. Health data use, stewardship, and governance: ongoing gaps and challenges: a report from AMIA's 2012 Health Policy Meeting. J Am Med Inform Assoc. 2013;21:204–11.
5. Lober WB, Thomas Karras B, Wagner MM, Marc Overhage J, Davidson AJ, Fraser H, et al. Roundtable on bioterrorism detection: information system–based surveillance. J Am Med Inform Assoc. 2002;9:105–15.
6. Lazarus R, Yih K, Platt R. Distributed data processing for public health surveillance. BMC Public Health. 2006;6:235.
7. El Emam K, Hu J, Mercer J, Peyton L, Kantarcioglu M, Malin B, et al. A secure protocol for protecting the identity of providers when disclosing data for disease surveillance. J Am Med Inform Assoc. 2011;18:212–7.
8. Lenert L, Sundwall DN. Public health surveillance and meaningful use regulations: a crisis of opportunity. Am J Public Health. 2012;102:e1–7.
9. Holmes JH, Elliott TE, Brown JS, Raebel MA, Davidson A, Nelson AF, et al. Clinical research data warehouse governance for distributed research networks in the USA: a systematic review of the literature. J Am Med Inform Assoc. 2014;21:730–6.
10. Finnell JT, Overhage JM, Grannis S. All health care is not local: an evaluation of the distribution of emergency department care delivered in Indiana. AMIA Annu Symp Proc. 2011;2011:409–16.
11. Gichoya J, Gamache RE, Vreeman DJ, Dixon BE, Finnell JT, Grannis S. An evaluation of the rates of repeat notifiable disease reporting and patient crossover using a health information exchange-based automated electronic laboratory reporting system. AMIA Annu Symp Proc. 2012;2012:1229–36.
12. Weber GM. Federated queries of clinical data repositories: the sum of the parts does not equal the whole. J Am Med Inform Assoc. 2013;20:e155–61.
13. Malin BA, El Emam K, O'Keefe CM. Biomedical data privacy: problems, perspectives, and recent advances. J Am Med Inform Assoc. 2013;20:2–6.
14. Laurie G, Jones KH, Stevens L, Dobbs C. A review of evidence relating to harm resulting from uses of health and biomedical data [Internet]. The Nuffield Council on Bioethics (NCOB); 2014 Jun p. 210. Available from: http://nuffieldbioethics.org/wp-content/uploads/FINAL-Report-on-Harms-Arising-from-Use-of-Health-and-Biomedical-Data-30-JUNE-2014.pdf
15. Du W, Atallah MJ. Privacy-preserving cooperative statistical analysis. In: Williams AD, editor. Comput. Secur. Appl. Conf. 2001 ACSAC 2001 Proc. 17th Annu. IEEE. 2001. p. 102–10.

16. Du W, Han YS, Chen S. Privacy-preserving multivariate statistical analysis: linear regression and classification. In: Berry MW, editor. Proc. Fourth SIAM Int. Conf. Data Min. SIAM. 2004. p. 222–33.
17. Kantarcioglu M. A survey of privacy-preserving methods across horizontally partitioned data. In: Aggarwal CC, Yu PS, editors. Priv.-Preserv. Data Min. New York: Springer; 2008. p. 313–35.
18. Vaidya J. A survey of privacy-preserving methods across vertically partitioned data. In: Aggarwal CC, Yu PS, editors. Priv.-Preserv. Data Min. New York: Springer; 2008. p. 337–58.
19. Clifton C, Kantarcioglu M, Vaidya J, Lin X, Zhu MY. Tools for privacy preserving distributed data mining. ACM SIGKDD Explor Newsl. 2002;4:28–34.
20. Hailemichael MA, Yigzaw KY, Bellika JG. Emnet: a tool for privacy-preserving statistical computing on distributed health data. In: Granja C, Budrionis A, editors. Proc. 13th Scand. Conf. Health Inform. Linköping: Linköping University Electronic Press; 2015. p. 33–40.
21. Andersen A, Yigzaw KY, Karlsen R. Privacy preserving health data processing. IEEE 16th Int. Conf. E-Health Netw. Appl. Serv. Heal. IEEE; 2014. p. 225–30
22. Vatsalan D, Christen P, Verykios VS. A taxonomy of privacy-preserving record linkage techniques. Inf Syst. 2013;38:946–69.
23. Pinkas B, Schneider T, Zohner M. Faster private set intersection based on OT extension. In: Fu K, Jung J, editors. Proc. 23rd USENIX Secur. Symp. San Diego: USENIX Association; 2014. p. 797–812.
24. Quantin C, Bouzelat H, Allaert FAA, Benhamiche AM, Faivre J, Dusserre L. How to ensure data security of an epidemiological follow-up:quality assessment of an anonymous record linkage procedure. Int J Med Inf. 1998;49:117–22.
25. Agrawal R, Evfimievski A, Srikant R. Information sharing across private databases. Proc. 2003 ACM SIGMOD Int. Conf. Manag. Data. New York, NY, USA: ACM; 2003. p. 86–97
26. El Emam K, Samet S, Hu J, Peyton L, Earle C, Jayaraman GC, et al. A protocol for the secure linking of registries for HPV surveillance. PLoS One. 2012;7: e39915.
27. Adam N, White T, Shafiq B, Vaidya J, He X. Privacy preserving integration of health care data. AMIA Annu. Symp. Proc. 2007. 2007. p. 1–5.
28. Lai PK, Yiu S-M, Chow KP, Chong CF, Hui LCK. An efficient bloom filter based solution for multiparty private matching. Secur. Manag. 2006. p. 286–292
29. Many D, Burkhart M, Dimitropoulos X. Fast private set operations with SEPIA. Technical report, ETH Zurich; 2012
30. Beimel A. Secret-sharing schemes: a survey. In: Chee YM, Guo Z, Shao F, Tang Y, Wang H, Xing C, editors. Coding Cryptol. Berlin: Springer; 2011. p. 11–46.
31. Dong C, Chen L, Wen Z. When private set intersection meets big data: an efficient and scalable protocol. Proc. 2013 ACM SIGSAC Conf. Comput. Commun. Secur. New York, NY, USA: ACM; 2013. p. 789–800
32. Kilian J. Founding crytpography on oblivious transfer. Proc. Twent. Annu. ACM Symp. Theory Comput. New York, NY, USA: ACM; 1988. p. 20–31.
33. Karapiperis D, Vatsalan D, Verykios VS, Christen P. Large-scale multi-party counting set intersection using a space efficient global synopsis. In: Renz M, Shahabi C, Zhou X, Cheema MA, editors. Database Syst. Adv. Appl. Springer International Publishing; 2015. p. 329–45.
34. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Stern J, editor. Adv. Cryptol. — EUROCRYPT'99. Berlin: Springer; 1999. p. 223–38.
35. Karr AF, Lin X, Sanil AP, Reiter JP. Secure regression on distributed databases. J Comput Graph Stat. 2005;14:263–79.
36. Bellika JG, Henriksen TS, Yigzaw KY. The Snow system - a decentralized medical data processing system. In: Llatas CF, García-Gómez JM, editors. Data Min. Clin. Med. Springer; 2014
37. Stewart BA, Fernandes S, Rodriguez-Huertas E, Landzberg M. A preliminary look at duplicate testing associated with lack of electronic health record interoperability for transferred patients. J Am Med Inform Assoc JAMIA. 2010;17:341–4.
38. Lazarus R, Kleinman KP, Dashevsky I, DeMaria A, Platt R. Using automated medical records for rapid identification of illness syndromes (syndromic surveillance): the example of lower respiratory infection. BMC Public Health. 2001;1:1.
39. Richesson RL, Horvath MM, Rusincovitch SA. Clinical research informatics and electronic health record data. Yearb Med Inform. 2014;9:215–23.
40. Curtis LH, Weiner MG, Boudreau DM, Cooper WO, Daniel GW, Nair VP, et al. Design considerations, architecture, and use of the Mini-Sentinel distributed data system. Pharmacoepidemiol Drug Saf. 2012;21:23–31.
41. Weber GM, Murphy SN, McMurry AJ, MacFadden D, Nigrin DJ, Churchill S, et al. The Shared Health Research Information Network (SHRINE): a prototype federated query tool for clinical data repositories. J Am Med Inform Assoc. 2009;16:624–30.
42. El Emam K, Mercer J, Moreau K, Grava-Gubins I, Buckeridge D, Jonker E. Physician privacy concerns when disclosing patient data for public health purposes during a pandemic influenza outbreak. BMC Public Health. 2011;11:454.
43. Lindell Y, Pinkas B. Secure multiparty computation for privacy-preserving data mining. J Priv Confidentiality. 2009;1:5.
44. Goldreich O. Secure multi-party computation (working draft). 2002. Available from http://www.wisdom.weizmann.ac.il/~oded/PSX/prot.pdf. Accessed 18 Oct 2016.
45. Cramer R, Damgård I. Multiparty computation, an introduction. In: Castellet M, editor. Contemp. Cryptol. Basel: Birkhäuser Basel; 2005. p. 41–87.
46. Goldreich O. Foundations of cryptography: basic applications. 1st ed. New York: Cambridge University Press; 2004.
47. Vaidya J, Clifton C. Leveraging the "Multi" in secure multi-party computation. Proc. 2003 ACM Workshop Priv. Electron. Soc. New York, NY, USA: ACM; 2003. p. 53–9
48. Bloom BH. Space/time trade-offs in hash coding with allowable errors. Commun ACM. 1970;13:422–6.
49. Tarkoma S, Rothenberg CE, Lagerspetz E. Theory and practice of bloom filters for distributed systems. Commun Surv Tutor IEEE. 2012;14:131–55.
50. Fan L, Cao P, Almeida J, Broder AZ. Summary cache: a scalable wide-area Web cache sharing protocol. IEEE ACM Trans Netw. 2000;8:281–93.
51. Dimitriou T, Michalas A. Multi-party trust computation in decentralized environments. 2012 5th Int. Conf. New Technol. Mobil. Secur. NTMS. 2012. p. 1–5
52. Dimitriou T, Michalas A. Multi-party trust computation in decentralized environments in the presence of malicious adversaries. Ad Hoc Netw. 2014;15:53–66.
53. Karr AF, Fulp WJ, Vera F, Young SS, Lin X, Reiter JP. Secure, privacy-preserving analysis of distributed databases. Technometrics. 2007;49:335–45.
54. Hernández MA, Stolfo SJ. Real-world data is dirty: data cleansing and the merge/purge problem. Data Min Knowl Discov. 1998;2:9–37.
55. Hernández MA, Stolfo SJ. The merge/purge problem for large databases. Proc. 1995 ACM SIGMOD Int. Conf. Manag. Data. New York, NY, USA: ACM; 1995. p. 127–38
56. Lunde AS, Lundeborg S, Lettenstrom GS, Thygesen L, Huebner J. The person-number systems of Sweden, Norway, Denmark, and Israel. Vital Health Stat 2. 1980;84:1–59.
57. Ludvigsson JF, Otterblad-Olausson P, Pettersson BU, Ekbom A. The Swedish personal identity number: possibilities and pitfalls in healthcare and medical research. Eur J Epidemiol. 2009;24:659–67.
58. Randall SM, Ferrante AM, Boyd JH, Semmens JB. The effect of data cleaning on record linkage quality. BMC Med Inform Decis Mak. 2013;13:64.
59. Sweeney L. Simple demographics often identify people uniquely [Internet]. Pittsburgh: Carnegie Mellon University; 2000 p. 1–34. Report No.: 3. Available from: http://dataprivacylab.org/projects/identifiability/paper1.pdf
60. El Emam K, Buckeridge D, Tamblyn R, Neisa A, Jonker E, Verma A. The re-identification risk of Canadians from longitudinal demographics. BMC Med Inform Decis Mak. 2011;11:46.
61. Koot M, Noordende G, Laat C. A study on the re-identifiability of Dutch citizens. Workshop Priv. Enhancing Technol. PET. 2010
62. Potosky AL, Riley GF, Lubitz JD, Mentnech RM, Kessler LG. Potential for cancer related health services research using a linked Medicare-tumor registry database. Med Care. 1993;31:732–48.
63. Warren JL, Klabunde CN, Schrag D, Bach PB, Riley GF. Overview of the SEER-Medicare data: content, research applications, and generalizability to the United States elderly population. Med Care. 2002;40:IV3–IV18.
64. Saint-Andre P, Smith K, Tronçon R. XMPP: the definitive guide: building real-time applications with jabber technologies. 1st ed. Sebastopol: O'Reilly Media, Inc.; 2009.
65. Collet Y. RealTime data compression: LZ4 explained [Internet]. 2011 [cited 2016 Apr 7]. Available from: http://fastcompression.blogspot.com/2011/05/lz4-explained.html
66. Friedman C, Rigby M. Conceptualising and creating a global learning health system. Int J Med Inf. 2013;82:e63–71.
67. Weber GM. Federated queries of clinical data repositories: scaling to a national network. J Biomed Inform. 2015;55:231–6.
68. Christen P. A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans Knowl Data Eng. 2012;24:1537–55.

69. Pohlig SC, Hellman ME. An improved algorithm for computing logarithms over and its cryptographic significance (Corresp.). IEEE Trans Inf Theory. 1978;24:106–10.

70. Schnell R, Bachteler T, Reiher J. Privacy-preserving record linkage using Bloom filters. BMC Med Inform Decis Mak. 2009;9:41.

71. Durham EA, Kantarcioglu M, Xue Y, Toth C, Kuzu M, Malin B. Composite bloom filters for secure record linkage. IEEE Trans Knowl Data Eng. 2014;26:2956–68.

72. Cormode G, Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications. J Algorithms. 2005;55:58–75.