The VLDB Journal (2006) 15:316–333
DOI 10.1007/s00778-006-0008-z

SPECIAL ISSUE PAPER

A secure distributed framework for achieving *k*-anonymity

Wei Jiang · Chris Clifton

Received: 30 September 2005 / Accepted: 25 May 2006 / Published online: 5 August 2006

Abstract *k*-anonymity provides a measure of privacy protection by preventing re-identification of data to fewer than a group of *k* data items. While algorithms exist for producing *k*-anonymous data, the model has been that of a single source wanting to publish data. Due to privacy issues, it is common that data from different sites cannot be shared directly. Therefore, this paper presents a two-party framework along with an application that generates *k*-anonymous data from two vertically partitioned sources without disclosing data from one site to the other. The framework is privacy preserving in the sense that it satisfies the secure definition commonly defined in the literature of Secure Multiparty Computation.

Keywords *k*-Anonymity · Privacy · Security

1 Introduction

Privacy is an important issue in our society and has become vulnerable in these technologically advanced times. Legislation has been proposed to protect individual privacy; a key component is the protection of individually identifiable data. Many techniques have been proposed to protect privacy, such as data perturbation [2], query restriction [7], data swapping [21], Secure Multi-party Computation (SMC) [11,35,36], etc. One challenge is relating such techniques to a privacy definition that meets legal and societal norms. Anonymous data are generally considered to be exempt from privacy rules – but what does it mean for data to be anonymous? Census agencies, which have long dealt with private data, have generally found that as long as data are aggregated over a group of individuals, release does not violate privacy. *k*-anonymity provides a formal way of generalizing this concept. As stated in [26,30,31], a data record is *k*-anonymous if and only if it is indistinguishable in its identifying information from at least *k* specific records or entities. The key step in making data anonymous is to generalize a specific value. For example, the ages 18 and 21 could be generalized to an interval [16–25]. Details of the concept of *k*-anonymity and ways to generate *k*-anonymous data are provided in Sect. 2.

Generalized data can be beneficial in many situations. For instance, a car insurance company may want to build a model to estimate claims for use in pricing policies for new customers. To build this model, the company may wish to use state-wide driver license records. Such records, even with name and ID numbers removed, are likely to contain sufficient information to link to an individual. However, by generalizing data (e.g., replacing a birth date with an age range [26–30]), it is possible to prevent linking a record to an individual. The generalized age range is likely to be sufficient for building the claim estimation model. Similar applications exist in many areas: medical research, education studies, targeted marketing, etc.

Due to vast improvements in networking and rapid increase of storage capacity, the full data about an individual are typically partitioned into several sub-data sets (credit history, medical records, earnings, ...), each

W. Jiang (✉) · C. Clifton
Purdue University, 305 N. University St.,
W. Lafayette, IN 47907-2107, USA
e-mail: wjiang@cs.purdue.edu

C. Clifton
e-mail: clifton@cs.purdue.edu

stored at an independent site.[1] The distributed setting is likely to remain, partially because of performance and accessibility, but more importantly because of autonomy of the independent sites. This autonomy provides a measure of protection for the individual data. For instance, if two attributes in combination reveal private information (e.g., airline and train travel records indicating likely attendance at political rallies), but the attributes are stored at different sites, a lack of cooperation between the sites ensures that neither is able to violate privacy.

In this paper, data are assumed to be vertically partitioned and stored at two sites, and the original data could be reconstructed by a one-to-one join on a common key. The goal is to build a *k*-anonymous join of the datasets, so that candidate keys in the joined dataset (including conjunction of items from different sites) are *k*-anonymized to prevent re-identification.

1.1 What is a secure distributed protocol?

The word "secure" under the context of this paper is related to the security definition of SMC. We also assume that parties are semi-honest in that they follow the execution requirement of the protocol but may use what they see during the execution to compute more than they need to know. (While the semi-honest model is a limitation, there are situations where a party will want to *avoid* knowing private data, to avoid the liability of protecting it – for such situations a semi-honest approach is appropriate.) Details of the security definitions and underlying models can be found in [12].

Definition 1 *Let T_i be the input of party i, $\prod_i(f)$ be the party i's execution image of the protocol f and s be the result computed from f. f is secure if $\prod_i(f)$ can be simulated from $\langle T_i, s\rangle$ and distribution of the simulated image is computationally indistinguishable from $\prod_i(f)$.*

While it has been shown that for any polynomial time algorithm, there exists a polynomial time secure protocol that achieves the same functionality, generic solutions presented in [11,36] require representing a problem as a boolean circuit. On a large dataset, it is computationally impractical to adopt the pure circuit-based generic solution. Therefore, the key challenge in designing a secure and efficient protocol is to avoid using large circuits, instead the generic approach must be used only for small circuits (i.e., simple functionalities with compact inputs).

[1] In the context of this paper, assume data are represented by a relational table, where each row indicates an individual data record and each column represents an attribute of data records.

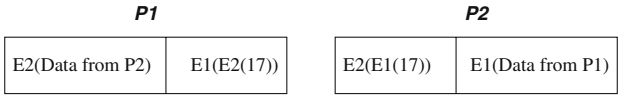


Fig. 1 Join on a global identifier without disclosing the identifier

1.2 Our contribution

There are two problems to be solved in constructing a *k*-anonymous dataset. The simple one is joining the data. At first glance, it may appear that each site could first *k*-anonymize its own dataset, then join on the (*k*-anonymous) candidate keys. However, this could lead to unintended consequences; for starters, a factor of *k* explosion in database size. More important, the resulting data could lead to wrong conclusions. Suppose that we *k*-anonymized a university employee dataset in this fashion, where one of the sites provided education level and other provided income (assume for the sake of the example that these are *not* deemed to be identifying information). Assume that the job classification was deemed to be identifying, and was anonymized so that work-study students and faculty were both anonymized to "university staff". In joining on the quasi-identifiers, the income of faculty would be joined to both the correct faculty and to students, leading to the (hopefully incorrect) conclusion that income is independent of education level.

A better solution is to join on a true global unique identifier, but without revealing that identifier (we assume the existence of such an identifier for this work). Commutative cryptography makes this feasible. A simple solution (based on [27]) is for each party to encrypt each record with its secret key, except for the global identifier. The global identifier is encrypted with its own key for the commutative encryption. The records are then given to the other party, which encrypts the (encrypted) global identifier with its commutative encryption key. Each party now has the (encrypted) records of the other party. Because of the commutativity of the encryption, the data can be joined on the (encrypted) global keys. Figure 1 shows how a record with key 17 would look at this stage; neither party knows what the key is or what the data is, but if the data are given to one party it can correctly join the records, as the keys are the same. However, since it knows only its own key, it is unable to learn the global identifier or the data from the other party.

Once the (encrypted) records are joined, the (encrypted) identifiers can be removed. Each party then decrypts their portion of the anonymous data, giving the anonymized dataset. Assuming semi-honest parties

(neither wants the responsibility of knowing the other's data), this protocol securely computes the join.[2]

Note that we have assumed that the records are anonymized before joining (except for the encrypted/removed key). This is not trivial. k-Anonymizing each dataset independently before joining does not guarantee a k-anonymous join. For example, Professor X may have "academic rank" as identifying information in one database, and "address" in the other. Rank may generalize to "faculty" and address to "third street". While there may be k faculty, and k university employees on third street, there may be only one faculty on third street, so the joined dataset is no longer k-anonymous.

The contribution of this paper is exactly the problem described above: How do we locally anonymize so that the joined dataset will be k-anonymous, without revealing anything in the process? We combine techniques from the cryptography community and k-anonymity to further enhance privacy in a distributed environment. This paper presents a two-party secure distributed framework that can be easily adopted to design a secure protocol to compute k-anonymous data from two vertically partitioned sources such that the protocol does not violate k-anonymity of either site's data. The framework is proven to reveal nothing to either party that cannot be derived from the k-anonymous dataset and the party's own data.

The key idea is that each party performs a local generalization as suggested by a regular k-anonymizing heuristic. The parties then test to see if the resulting join will be k-anonymous, without revealing anything except if the data are sufficiently anonymized. If not, each party then generalizes further. This continues until the data are sufficiently anonymized, at which point the process described above could be used to join the datasets.

We first introduce the fundamental concepts of k-anonymity. Section 3 gives a secure set intersection protocol that we will use as a subroutine in our framework; this also serves as a refresher on SMC. Section 4 presents the secure distributed framework, with proof of its correctness, privacy-preservation property, and a discussion of the computational and communication complexity. Section 5 presents an application of such framework where we show how to utilize it to produce a secure distributed k-anonymization protocol. Section 6 provides empirical results that demonstrates the effectiveness of the proposed framework in creating a secure distributed k-anonymization protocol. The paper concludes with some insights gained from the framework and its application and future research directions on achieving k-anonymity in a distributed environment.

[2] Since we do not claim this is an original contribution, we leave the rather straightforward proof as an exercise to the reader.

2 Related work and background

Several centralized algorithms have been proposed to generate k-anonymous data, such as those proposed in [3,15,19,28,30]. Two distributed protocols were proposed in [37], one of which is to extract k-anonymous portion of the data and other is the secure version of the protocol presented in [20]. Both protocols assume the data is horizontally partitioned. The targeted problem of this paper is similar to the problem addressed in [18]. The previous protocol, however, does not satisfy the security definition commonly adopted in the literature of SMC. Although the approach in [18] was shown to maintain k-anonymity among the participating parties, it potentially reveals alternate k-anonymous datasets that give the parties more information than in the final dataset. In contrast, the protocol proposed in this paper meets definitions from SMC, thus ensuring nothing is revealed except the final k-anonymous dataset.

2.1 k-Anonymity

We now give key background on k-anonymity, including definitions, a single-site algorithm, and a relevant theorem, from [26,29,31]. The following notations are crucial for understanding the rest of the paper:

- Quasi-identifier (QI): a set of attributes that can be used with certain external information to identify a specific individual.
- $T, T[QI]$: T is the original dataset represented in a relational form, $T[QI]$ is the projection of T to the set of attributes contained in QI.
- $T_k[QI]$: k-anonymous data generated from T with respect to the attributes in the Quasi-identifier QI.

Definition 2 *$T_k[QI]$ satisfies k-anonymity if and only if each record in it appears at least k times.*

Let T be Table 1, T_k be Table 2 and QI = {AREA, POSITION, SALARY}. According to Definition 2, $T_k[QI]$ satisfies 3-anonymity. (Note that the SSN field is unique, but since it is not a part of QI, this does not violate the definition of k-anonymity. In operational terms, we are assuming that an adversary does not know or care about the value of SSN, so disclosing it does not enable reidentification or otherwise pose a privacy risk. In practice, a global identifier such as SSN would be used and removed in the join process described on Page 2. For

Algorithm 1 Key Steps in Datafly

Require: $T, QI[A_1, \ldots, A_m], k$, Hierarchies VGHs, Assume $k \leq |T|$

1: $freq \leftarrow$ a frequency list contains distinct sequences of values of $T[QI]$ along with the number of occurrences of each sequence.
2: **while** $(\exists s \in freq \wedge occur(s) < k)$ **do**
3: $A_i \in QI$ having the most number of distinct values
4: $freq \leftarrow$ generalize the values of $A_i \in freq$
5: **end while**
6: $T_k[QI] \leftarrow$ construct table from $freq$
7: *return* $T_k[QI]$

clarity, in the examples, we will use the ID in column 1 rather than SSN as a global identifier.)

Datafly [28,30] is a simple and effective algorithm, so for demonstration of our protocol, Datafly is used to make local data k-anonymous. Algorithm 1 presents the key steps in Datafly. The main step in most k-anonymity protocols is to substitute a specific value with a more general value. For instance, Fig. 2a contains a value generalization hierarchy (VGH) for attribute AREA, in which Database Systems is a more general value than Data Mining. Similarly, Fig. 2b, c presents VGHs for attributes POSITION and SALARY contained in QI. Continuing from the previous example, $T_k[QI]$ of Table 2 satisfies 3-anonymity. According to the three VGHs and the original data represented by T, it is easily verified that Datafly could generate $T_k[QI]$ by generalizing the data on SALARY, then on AREA and then on SALARY again. Next, we present a useful theorem about k-anonymity.

Theorem 1 *If $T_k[QI]$ is k-anonymous, then $T_k[QI']$ is also k-anonymous, where $QI' \subseteq QI$ [30].*

Proof Assume $T_k[QI]$ is k-anonymous and $T_k[QI']$ does not satisfy k-anonymity. Then there exists a record $t(QI')$ that appears in $T_k[QI']$ less than k times. It is trivial to observe that $t(QI)$ also appears less than k times in $T_k[QI]$. That contradicts the assumption. Therefore, if $T_k[QI]$ satisfies k-anonymity, so does $T_k[QI']$. □

3 Secure set intersection

We now present a secure two-party protocol SSI_t for computing the cardinality of set intersection that takes a random bit as part of input from one party and returns another random bit to the other party; the XOR of the two bits indicates whether or not the size of the intersection size is smaller or greater than a threshold t. SSI_t will be used to test if the join of two local k-anonymous datasets creates a global k-anonymization, and this test plays an important role in the design of our secure distributed k-anonymity framework. We provide a concrete construction of SSI_t here, and the construction also serves as an introduction to basic concepts (e.g., proof of security) in the literature of SMC.

Most existing secure set intersection protocols, such as those proposed in [1,9,33], disclose the size of each party's dataset (or a bound on that size). For certain applications, this disclosure is acceptable. In our application, such a disclosure would prevent a protocol from meeting the definitions of SMC, causing a leak with unknown implications (this is discussed further in Sect. 4). Although the protocols proposed in [10] and [34] meet the above requirement, they do not satisfy the requirement that the results be randomly split between parties. As we shall see, we need to learn not the size of single set intersection, but if the sizes of all exceed the threshold k. The SSI_t protocol presented below utilizes the same structure as the ones presented in [10] and [34], except that the SSI_t protocol also adopts an additional circuit to produce two random bits.

SSI_t merely discloses the domain size, and according to the k-anonymity application investigated in this paper, the domain size is bounded by the size of database. Therefore, the domain size is generally much smaller than the key size adopted in an encryption scheme. For better understanding and completeness, we provide detailed description of SSI_t in the rest of this section.

3.1 SSI_t

Let D_1, D_2 be the datasets (containing integer values) of parties P1 and P2, respectively. Let M be maximum value in the domain from which the elements of the two datasets are drawn. Then we generate the corresponding representations s_1, s_2 of D_1, D_2, where s_1, s_2 are in forms of bit strings. Let $s_j[i]$ ($j = 1$ or 2) indicates the ith bit of s_j, and the s_j has the property that:

$|s_j| = M$ (the length of the bit string s_j); $s_j[i] = 1$ if $i \in D_j$, otherwise $s_j[i] = 0$.

Formally, define $\text{SSI}_t((s_1, t), (s_2, t, b_2)) \rightarrow (b_1, \perp)$, where t is publicly known threshold, b_1, b_2 are two random bits (without loss of generality, assume b_2 is provided as part of input from P2), (s_1, t) is P1's input and (s_2, t, b_2) is P2's input. After the execution of SSI_t, P1 receives a random bit b_1, and P2 receives an empty string indicated by the symbol $\perp$. If $|D_1 \cap D_2| < t, b_1 \oplus b_2 = 1$; otherwise $b_1 \oplus b_2 = 0$.

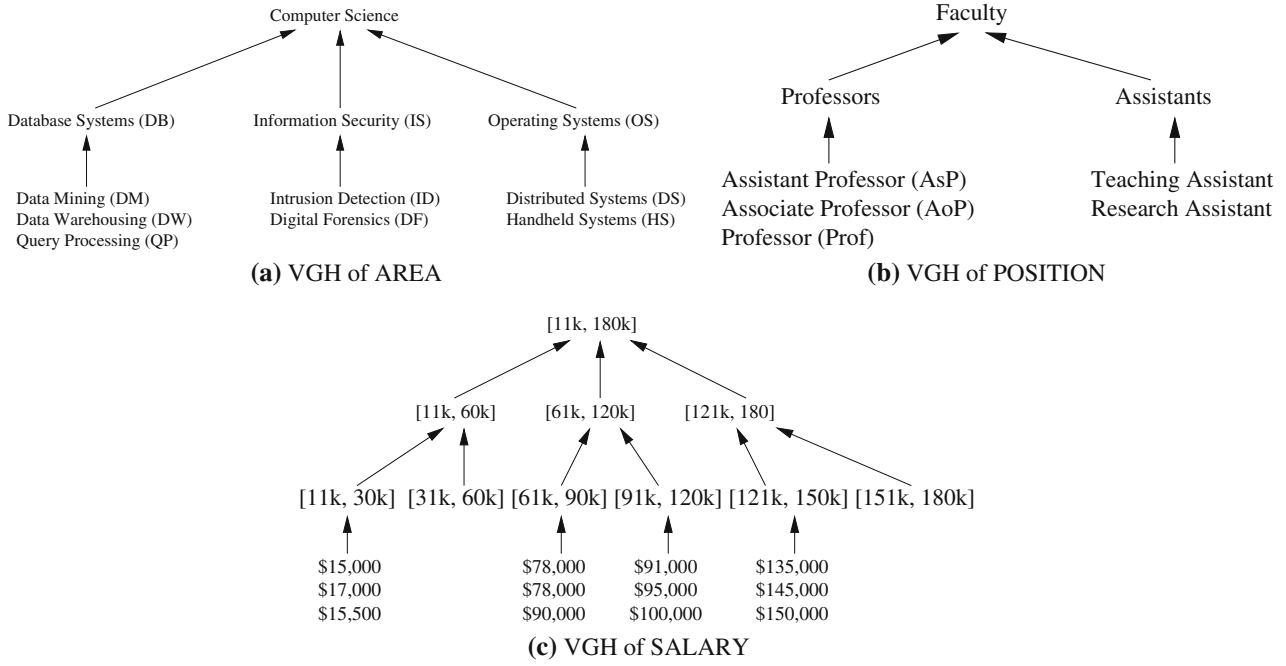
SSI_t is an extension of the Boolean dot product protocol described in [32]. The bit strings s_1, s_2

Table 1 Original dataset before partitioning

ID	Area	Position	Salary	SSN
1	Data mining	Associate professor	$90,000	708-79-1698
2	Intrusion detection	Assistant professor	$91,000	606-67-6789
3	Data warehousing	Associate professor	$95,000	626-23-1459
4	Intrusion detection	Assistant professor	$78,000	373-55-7788
5	Digital forensics	Professor	$150,000	626-87-6503
6	Distributed systems	Research assistant	$15,000	708-66-1552
7	Handhold systems	Research assistant	$17,000	810-74-1079
8	Handhold systems	Research assistant	$15,500	606-37-7706
9	Query processing	Associate professor	$100,000	373-79-1698
10	Digital forensics	Assistant professor	$78,000	999-03-7892
11	Digital forensics	Professor	$135,000	708-90-1976
12	Intrusion detection	Professor	$145,000	606-17-6512

Table 2 Generalized data with $k = 3$

ID	Area	Position	Salary	SSN
1	Database systems	Associate professor	[61k, 120k]	708-79-1698
2	Information security	Assistant professor	[61k, 120k]	606-67-6789
3	Database systems	Associate professor	[61k, 120k]	626-23-1459
4	Information security	Assistant professor	[61k, 120k]	373-55-7788
5	Information security	Professor	[121k, 180k]	626-87-6503
6	Operating systems	Research assistant	[11k, 30k]	708-66-1552
7	Operating systems	Research assistant	[11k, 30k]	810-74-1079
8	Operation systems	Research assistant	[11k, 30k]	606-37-7706
9	Database systems	Associate professor	[61k, 120k]	373-79-1698
10	Information security	Assistant professor	[61k, 120k]	999-03-7892
11	Information security	Professor	[121k, 180k]	708-90-1976
12	Information security	Professor	[121k, 180k]	606-17-6512



(a) VGH of AREA

(b) VGH of POSITION

(c) VGH of SALARY

Fig. 2 Value generalization hierarchies

can be represented as Boolean vectors whose entries contain either 0 or 1. Since the dot product of two Boolean vectors results the summation (*sum*) of 1's remaining after bit-wise multiplication of the two vectors, $sum = |D_1 \cap D_2|$. Let *sum* denote the intersection size of two datasets for the rest of the section.

SSI_t is a generic protocol in that any homomorphic probabilistic public key encryption systems, such as those proposed in [4,22,24], can be adopted in its implementation. Let $E : R \times X \rightarrow Y$ be a probabilistic public key encryption scheme, where R, X and Y are finite domains identified with an initial subset of integers and $D : Y \rightarrow X$ be a private decryption algorithm, such that $\forall(r,x) \in R \times X, D(E(r,x)) = x$. Furthermore, these encryption systems have the following properties:

- The encryption function is injective with respect to the second parameter, i.e., $\forall(r_1,x_1),(r_2,x_2) \in R \times X$, $E(r_1,x_1) = E(r_2,x_2) \Rightarrow x_1 = x_2$.
- The encryption function is additive homomorphic, i.e., $\forall(r_1,x_1),(r_2,x_2) \in R \times X, \prod(E(r_1,x_2),E(r_2,x_2)) = E(r_3,x_1+x_2)$, where r_3 can be computed from r_1, r_2, x_1 and x_2 in polynomial time. ($\prod$ is the function to "add" two encrypted values; multiplication in the systems listed above.)
- The encryption function has semantic security as defined in [14]. Informally speaking, a set of ciphertexts does not provide additional information about the plaintext to an adversary with polynomial-bounded computing power.
- The domain and the range of the encryption system are suitable.

Key steps of the SSI_t protocol are highlighted in Algorithm 2. At step 2(a), the symbol $\prod$ indicates that the encrypted $s_1[i]$ values are combined to produce their encrypted *sum*; the common characteristics among all these values is that their corresponding $s_2[i]$ values must be 1. At step 2(b), an encrypted sum-share is computed for P1. The symbol SH_{P2}^{-1} at step 4(a) represents the inverse of SH_{P2} so that $SH_{P1} + SH_{P2}^{-1} = sum + SH_{P2} + SH_{P2}^{-1} = sum$. At step 4, the two parties engage in a simple secure circuit evaluation process.

The circuit *Secure_COM* is defined in Algorithm 3. *Secure_COM* takes a random SH_{P1} from P1 and a random share SH_{P2}^{-1} and random bit b_2 from P2. It returns a random bit b_1 to P1 such that if $b_1 \oplus b_2 = 1$, then $SH_{P1} + SH_{P2}^{-1} < t$ (t is a publicly known threshold); otherwise, $SH_{P1} + SH_{P2}^{-1} \geq t$. Note that at step 1 of Algorithm 3, *ADD* indicates an addition circuit that takes two integers (in the same group or field with modulo operation over the size of the group or field) as input, and at

Algorithm 2 SSI_t Protocol

Require: $(s_1,t),(s_2,t,b_2),|s_1| = |s_2| = M$

1: Party 1 (P1):
   (a). generate a public and private key pair E, D for the probabilistic public key homomorphic encryption system.
   (b). encrypt s_1:
   $x_i \leftarrow E(r, s_1[i])$, for $i = 1,\ldots,M$; r is randomly chosen for each $s_1[i]$
   (c). send $x_1,\ldots,x_M$ in order and E to P2
2: Party 2 (P2):
   (a). compute $Enc\_Sum \leftarrow \prod_{\forall i \wedge s_2[i]=1} x_i$
   (b). compute $Enc\_SumSh \leftarrow \prod(Enc\_Sum, E(r', SH_{P2}))$, where r' and SH_{P2} are randomly chosen
   (c). send Enc_SumSh to P1
3: Party 1:
   (a). compute $SH_{P1} \leftarrow D(Enc\_SumSh)$
4: Party 1 & Party 2
   (a). $Secure\_COM((SH_{P1},t),(SH_{P2}^{-1},t,b_2))$
   (b). P1 $\leftarrow b_1$ & P2 $\leftarrow \perp$

Algorithm 3 Secure_COM Protocol

Require: $((sh1,t),(sh2,t,b_2))$, where t is a threshold and b_2 is a random bit

1: $s = ADD(sh1, sh2)$
2: $b_1 = COMPARE(s,t) \oplus b_2$
3: return $(b_1, \perp)$

step 2, $COMPARE(s,t)$ indicates a comparison circuit that takes two integers as input and returns 1 if $s < t$ and 0 otherwise. For succinctness and clarity, we present this *Secure_COM* circuit as two sub-circuits: *ADD* and *COMPARE*. However, we securely evaluate them as a whole via the approach presented in [12]. Therefore, the intermediate results computed by *ADD* and *COMPARE* are not shared.

As stated previously, except for this additional circuit *Secure_COM*, SSI_t has the same underlying structure as those protocols proposed in [10] and [34]. In addition, SSI_t (Algorithm 2) has been optimized at step 2(a): multiplications are required only when $s_2[i] = 1$.

A similar protocol PM_t has been proposed in [9]. PM_t cannot be used directly because it discloses the sizes of the datasets. Moreover, in addition to the secure root finding polynomial evaluations, a circuit is also required in PM_t to sequentially compare elements in each dataset. Although the SSI_t protocol requires a circuit to return two random bits, the complexity of this circuit is similar to a simple circuit comparing two numbers. As a result, the circuit used in SSI_t is much simpler than that in PM_t.

Since SSI_t is merely related to values of individual bits under the context of this paper, it can be directly

implemented by circuits. We provide such a circuit in the Appendix. The circuit presented in Algorithm 8 may not be the most efficient one, but it is simple and presents an appropriate upper bound on any purely circuit-based implementation of the SSI_t protocol. In this paper, we prefer homomorphic encryption based implementation of SSI_t, in part because homomorphic encryption is easy to implement and in part because secure circuit evaluation is a complex process. Although the implementation of SSI_t presented in this section requires a comparison circuit, it is much smaller comparing to the one illustrated in Algorithm 8. Also, the two different implementations of SSI_t have similar complexity.

3.2 Proof of correctness

Theorem 2 *After each execution of the SSI_t protocol defined in Algorithm 2, P1 and P2 have two random bits b_1 and b_2, such that if sum $= |D_1 \cap D_2| < t$, $b_1 \oplus b_2 = 1$; otherwise, $b_1 \oplus b_2 = 0$, where D_1 and D_2 are datasets (containing only integer values) of P1 and P2.*

Proof Because the multiplication of bit 0 with any other bit does not contribute to the total sum, P2 only uses x_i's whose corresponding $s_2[i]$ values are 1. Thus, step 2(a) of Algorithm 2 computes the encrypted *sum*: $Enc\_Sum = \prod_{\forall i \wedge s_2[i]=1} x_i = E(r, \sum_{\forall i \wedge s_2[i]=1} s_1[i]) = E(r', \sum_{i=1,\ldots,M} s_1[i] \cdot s_2[i])$, for some random numbers r, r'.

When *Secure_COM* is securely evaluated, SH_{P1} and SH_{P2}^{-1} are the actual parameters. In other words, during the evaluation of *Secure_COM*, $sh1 = SH_{P1}$ and $sh2 = SH_{P2}^{-1}$. As explained in the previous section, $s = sh1 + sh2 = SH_{P1} + SH_{P2}^{-1} = sum$. As a result, s contains the exact value of the intersection size. In addition, in part because $COMPARE(s, t)$ returns 1 if $s < t$ and 0 otherwise and in party because $b_1 = COMPARE(s, t) \oplus b_2$, $b_1 \oplus b_2 = COMPARE(s, t)$ which presents the precise comparison results between the intersection size and the threshold. This concludes that Algorithm 2 correctly implements the SSI_t protocol. □

3.3 Proof of security

Theorem 3 *The SSI_t protocol defined in Algorithm 2 is secure under the semi-honest definition of SMC.*

Proof The proof is based on simulation. We must define a simulator that can take as input one party's data and the final result, and generate a stream of messages that are computationally indistinguishable from those the party would receive in an actual run of the protocol. Indistinguishable means the distribution of messages across multiple runs with the same input and final result; since (randomly chosen) encryption keys may lead to different actual values, the messages may not be exactly the same. However, the distributions are the same: every message stream produced by the simulator is equally likely to be the one actually seen in the protocol (for details see [12]).

At step 4 of Algorithm 2, P1 and P2 collaboratively evaluate the circuit *Secure_COM*. The circuit can be evaluated securely based on the method defined in [11, 12]. Based on the composition theorem of [12], it is sufficient to define a simulator for the messages received by each party in steps 1 – 3 of Algorithm 2.

Simulator for P1: P1 only receives $M_{P1} = E(r, SH_{P1})$. Since P1 obtains SH_{P1} after the decryption, P1 can generate a $M'_{P1} = E(r', SH_{P1})$. The encryption scheme is semantically secure, so M_{P1} and M'_{P1} are computationally indistinguishable.

Simulator for P2: The public key E received by P2 can be simulated by randomly select a key from the key space. Because $x_1, \ldots, x_M$ are encrypted values of 0 or 1, P2 can simulate each x_i value as follows: first P2 randomly selects an encryption key, a number r and a *bit* (0 or 1), and then P2 computes $x'_i = E(r, bit)$. The semantic security of the encryption scheme guarantees that x_i and x'_i are computationally indistinguishable. Similar proofs can be found in [32]; a definition of computational indistinguishability can be found in [13]. □

3.4 Complexity of SSI_t

We now show the communication and encryption costs of SSI_t. The costs are in terms of M, the domain size that bounds the size of the dataset, and N, the domain size from which private–public key pairs are drawn (i.e., $\log M$ or $\log N$ is the maximum number of bits needed to represent a value from its corresponding domain).

Theorem 4 *The number of encryptions and the communication complexity of the SSI_t protocol are bounded by $O(M)$ and $O(M \log N)$, respectively.*

Proof Since the length of each bit string s_i is M, the number of encryption for P1 is M at step 1 of Algorithm 2. At step 2(b), P2 needs to perform 1 encryption and at most $M + 1$ multiplications (assuming a homomorphic encryption scheme where multiplying encrypted values is used to add the underlying values). In addition, P1 performs 1 decryption at step 3. Thus, because the size of an encrypted value is $\log N$ in number of bits, the number of encryptions and communication complexity from steps 1 to 3 of Algorithm 2 are bounded by $O(M)$ and $O(M \log N)$ (in bits), respectively.

The complexity of the *Secure_COM* protocol is determined by both ADD and $COMPARE$ circuits. Assume

Table 3 P1 and P2 's generalized data (left and right, respectively)

ID	Area[1]	Position[0]	Area[1]	Position[1]
1	DB	AoP	DB	Professors
2	IS	AsP	IS	Professors
3	DB	AoP	DB	Professors
4	IS	AsP	IS	Professors
5	IS	Prof	IS	Professors
6	OS	RA	OS	Assistant
7	OS	RA	OS	Assistant
8	OS	RA	OS	Assistant
9	DB	AoP	DB	Professors
10	IS	AsP	IS	Professors
11	IS	Prof	IS	Professors
12	IS	Prof	IS	Professors

ID	Salary[1]	Salary[2]
1	[61k, 90k]	[61k, 120k]
2	[91k, 120k]	[61k, 120k]
3	[91k, 120k]	[61k, 120k]
4	[61k, 90k]	[61k, 120k]
5	[121k, 150k]	[121k, 180k]
6	[11k, 30k]	[11k, 30k]
7	[11k, 30k]	[11k, 30k]
8	[11k, 30k]	[11k, 30k]
9	[91k, 120k]	[61k, 120k]
10	[61k, 90k]	[61k, 120k]
11	[121k, 150k]	[121k, 180k]
12	[121k, 150k]	[121k, 180k]

ADD is a sequential adder. The number of gates for such adder is generally bounded by $O(\log N)$ since the value of each random share is bounded by N. Also, the number of gates for the comparison circuit is bounded by $O(\log M)$ (a linear circuit in number of bits). Therefore, the circuit *Secure_COM* requires $O(\log N)$ gates (assuming $N > M$). Based on the secure circuit evaluation approach stated in [11], the number of encryptions to encrypt *Secure_COM* is bounded by its size: $O(\log N)$. As a consequence, the communication complexity of *Secure_COM* is bounded by $O(\log N \log N)$ (in bits), where $\log N$ is the size of the encryption and decryption keys.

According to the above analysis and assuming $M \gg \log N$, the number of encryptions and the communication complexity of the SSI_t protocol are bounded by $O(M)$ and $O(M \log N)$, respectively. □

4 The framework: DkA

We now present a secure distributed framework DkA (Distributed k-Anonymity) for generating a global k-anonymization between two parties. DkA cannot be directly used to compute a global k-anonymous dataset since it is a framework and does not specify how to anonymize data. On the other hand, it can be easily adopted to construct a secure protocol to achieve global k-anonymization between two parties. In Sect. 5, we present an application that shows how to incorporate one of existing k-anonymization algorithms into this framework.

The rest of the section is organized as the following: The framework is presented in Sect. 4.1. Section 4.2 proves the correctness of the framework (the result is guaranteed to be k-anonymous), and Section 4.3 proves the framework satisfies the security definition in the SMC literature.

The following symbols are used extensively in this section: γ_c^i, θ_c^i and ω_c^i, where the superscript $i \in \{1, 2\}$ indicates whether the symbol represents information from P1 or from P2, and the subscript $c \in$ *Integer* indicates the protocol round generating the information. For instance, γ_1^1 and γ_2^1 represent values that P1 has during rounds 1 and 2. The meaning of the symbols is defined as the framework is introduced.

4.1 DkA

We will present the framework using the example given earlier. The framework is executed between two parties: P1 and P2. For illustration purpose, assume both parties agree on a k-anonymization algorithm that they use to produce locally k-anonymous datasets. Let T refer to Table 1 and QI = {AREA, POSITION, SALARY}. T is vertically partitioned into $T1$[ID, AREA, POSITION] and $T2$[ID, SALARY, SSN] stored at P1 and P2, respectively. Also, assume P1 and P2 are semi-honest in that they follow the execution of the framework but may later use the information seen to try to violate privacy.

The key idea of the framework is based on Theorem 1. Initially, each party Pi k-anonymizes its local data (for simplicity, Datafly is used for illustration). Based on this locally k-anonymous data, a set γ_c^i is produced containing IDs partitioned into subsets. Let $\gamma_c^i[p]$ indicates the pth subset in γ_c^i, then all records of Pi whose keys are contained in $\gamma_c^i[p]$ have the same value with respect to QI. $\forall \gamma_c^i$, the following properties hold:

- $\gamma_c^i[p] \cap \gamma_c^i[q] = \emptyset, \forall p, q$ s.t. $1 \le p, q \le |\gamma_c^i|$ and $p \ne q$
- $\bigcup_{1 \le p \le |\gamma_c^i|} \gamma_c^i[p]$ is the same across all c values

(Note that although each element $\gamma_c^i[p] \in \gamma_c^i$ contains record keys, it does make sense to say that $\gamma_c^i[p]$

contains a subset of records or data tuples because each key is related to a single tuple). Define $Ti_{\gamma_c^i}$ to be the generalized or anonymous data at Pi based on which γ_c^i is computed. For an example, refer to Table 3. The columns [AREA^p, POSITION^q] indicate the generalized data of $T1$[AREA, POSITION], where $p+q$ indicates the number of times $T1$[AREA, POSITION] has been generalized (by Datafly). Also, the last generalization of $T1$[AREA, POSITION] was performed on the attribute whose superscript was incremented.

$T2$[SALARY] can be interpreted similarly. According to Table 3, we have:

$$\gamma_1^1 = \{\{1,3,9\},\{2,4,10\},\{5,11,12\},\{6,7,8\}\},$$
$$\gamma_1^2 = \{\{1,4,10\},\{2,3,9\},\{5,11,12\},\{6,7,8\}\}.$$

The two parties then compare γ_1^1 and γ_1^2. If they are *anonymized* (this notion of being *anonymized* will be defined shortly), joining data $T1_{\gamma_1^1}$ and $T2_{\gamma_1^2}$ creates globally k-anonymous data. If γ_1^1 and γ_1^2 are *not-anonymized*, each party generalizes its local data one step further (assume they know how to do this) and creates a new γ_c^i. Repeat the above steps until the two parties find a pair *anonymized* γ_c^is. We now define the notion of being *anonymized* between any two γ_c^is.

Definition 3 *(*Anonymized*) If $\gamma_c^1 \equiv \gamma_c^2$, then there are no p,q such that $0 < |\gamma_c^1[p] \cap \gamma_c^2[q]| < k$. ($\gamma_c^1 \equiv \gamma_c^2$ indicates γ_c^1 and γ_c^2 are anonymized, and $\gamma_c^1 \neq \gamma_c^2$ indicates the opposite: not-anonymized)*

According to the above definition, $\gamma_1^1 \neq \gamma_1^2$ because $|\{1,3,9\} \in \gamma_1^1 \cap \{2,3,9\} \in \gamma_1^2| = 2 < k(=3)$. Thus, P1 and P2 generalize their data one step further and compute two new γ_c^is:

$$\gamma_2^1 = \{\{1,3,9\},\{2,4,5,10,11,12\},\{6,7,8\}\},$$
$$\gamma_2^2 = \{\{1,2,3,4,9,10\},\{5,11,12\},\{6,7,8\}\}.$$

Since $\gamma_2^1 \equiv \gamma_2^2$, the join of $T1_{\gamma_2^1}$ (columns [AREA^1, POSI-TION^1] in Table 3) and $T2_{\gamma_2^2}$ (column [SALARY^2] in Table 3) satisfies 3-anonymity.

The idea is to check if each identifier j is part of a k-anonymous set in the joined database; the intersection of sets containing j is exactly the set of items with quasi-identifiers equal to those of j. To prevent information leakage, the comparison or the anonymity test between any two γ_c^is is not performed directly. Instead, the comparison process merely enables the two parties to know the result of the anonymity test, but not the specific subsets of γ_c^is for which the anonymity test fails. Before detailing the secure anonymity test between γ_c^is, let t_j indicate a record or tuple in the database whose ID attribute has value j. Assume S is a set. Let $Clone_z(S)$ indicate the set S is copied z times, and assume M is the size of the domain (or the maximum value in the domain) of the dataset. Based on the value of γ_c^is, each party Pi computes a $\theta_c^i = \{\forall \gamma_c^i[j] \in \gamma_c^i, Clone_{|\gamma_c^i[j]|}(\gamma_c^i[j])\}$, where the cardinality of θ_c^i is M. Call θ_c^i the "by identifier" subset list.

Elements in each θ_c^i are sorted according to value j of t_j or the ID values. In other words, the jth subset in θ_c^i must contain the ID value j. For example, according to γ_1^1, γ_1^2, P1 and P2 compute θ_1^1, θ_1^2:

$$\begin{aligned}\theta_1^1 = (&\{1,3,9\},\{2,4,10\},\{1,3,9\},\{2,4,10\},\{5,11,12\},\\ &\{6,7,8\},\{6,7,8\},\{6,7,8\},\{1,3,9\},\{2,4,10\},\\ &\{5,11,12\},\{5,11,12\}),\\ \theta_1^2 = (&\{1,4,10\},\{2,3,9\},\{2,3,9\},\{1,4,10\},\{5,11,12\},\\ &\{6,7,8\},\{6,7,8\},\{6,7,8\},\{2,3,9\},\{1,4,10\},\\ &\{5,11,12\},\{5,11,12\}).\end{aligned}$$

Next, P1 and P2 compute the required parameters for the SSI$_t$ protocol described in Sect. 3 before applying SSI$_t$ to each pair $\theta_1^1[j], \theta_1^2[j]$. Note that the threshold parameter to SSI$_t$ is always k (where $k=3$ in our example). In addition, P2 is responsible for supplying a randomly chosen bit b_2 for each execution of SSI$_t$, and each execution of SSI$_t$ returns a bit b_1 to P1. Because of the construction of θ_c^i, $0 < |\theta_c^1[j] \cap \theta_c^2[j]|$ for $1 \leq j \leq M$; therefore, the following holds: if $b_1 \oplus b_2 = 1$, $0 < |\theta_c^1[j] \cap \theta_c^2[j]| < k$; otherwise, $|\theta_c^1[j] \cap \theta_c^2[j]| \geq k$. After performing the SSI$_t$ protocol on each pair $\theta_1^1[j]$, $\theta_1^2[j]$, P2 has a string ω_1^2 of random bits (each bit is randomly generated by P2). P1 obtains a string ω_1^1 of bits that from P1's point of view are indistinguishable from random. For instance, based on θ_1^1, θ_1^2 and supposing P2 randomly generates $\omega_1^2 = 100111011110$, then after collaboratively executing the SSI$_t$ protocol on the M pair $\theta_1^1[j], \theta_1^2[j]$s, P1 get $\omega_1^1 = 011011010010$. ($M = 12$ in this example.)

P1 and P2 then securely compare ω_1^1 and ω_1^2. If the two bit strings are equal, the two corresponding γ_1^1 and γ_1^2 are *anonymized* (according to Definition 3). We will formally prove this claim in Sect. 4.2. Since ω_1^1 and ω_1^2 are not equal in the current example, P1 and P2 compute θ_2^1, θ_2^2 based on γ_2^1, γ_2^2:

$$\begin{aligned}\theta_2^1 = (&\{1,3,9\},\{2,4,5,10,11,12\},\{1,3,9\},\\ &\{2,4,5,10,11,12\},\{2,4,5,10,11,12\},\\ &\{6,7,8\},\{6,7,8\},\{6,7,8\},\{1,3,9\},\\ &\{2,4,5,10,11,12\},\{2,4,5,10,11,12\},\\ &\{2,4,5,10,11,12\}),\end{aligned}$$

Algorithm 4 DkA (shown for P1; P2 identical).

Require: Private Data $T1$, QI_{P1} (a subset of QI related to P1), Constraint k, $VGH_{A_i}, \forall A_i \in QI_{P1}$, assume $k \leq |T1| = M$

1: P1 generalizes $T1$ to be locally k-anonymous;
2: int $c \leftarrow 0$;
3: **repeat**
4: $c = c + 1$;
5: P1 computes k-anonymous subsets γ_c^1;
6: P1 computes θ_c^1, the "by identifier" subset list;
7: P1 obtains the "by identifier" comparison shares ω_c^1 by collaboratively executing SSI_t once on each pair $\theta_c^1[j], \theta_c^2[j]$, for $1 \leq j \leq M$;
8: **until** $Is\_Equal(\omega_c^1, \omega_c^2)$;
9: *return* $T_k[QI] \leftarrow T1_{\gamma_c^1} \bowtie T2_{\gamma_c^2}$;

$$\theta_2^2 = (\{1,2,3,4,9,10\}, \{1,2,3,4,9,10\}, \{1,2,3,4,9,10\}, \{1,2,3,4,9,10\}, \{5,11,12\}, \{6,7,8\}, \{6,7,8\}, \{6,7,8\}, \{1,2,3,4,9,10\}, \{1,2,3,4,9,10\}, \{5,11,12\}, \{5,11,12\}).$$

Suppose P2 generates another random bit string $\omega_2^2 = 010011010110$, then by sequentially executing SSI_t on each pair $\theta_2^1[j], \theta_2^2[j]$, P1 obtains the corresponding random bit string $\omega_2^1 = 010011010110$. After performing a secure equality test that confirms $\omega_2^1 = \omega_2^2$, the two parties can conclude $\gamma_2^1 \equiv \gamma_2^2$, and as stated previously, the join of $T1_{\gamma_2^1}$ and $T2_{\gamma_2^2}$ creates a globally 3-anonymous dataset.

The key steps in our framework are highlighted in Algorithm 4. The algorithm is written as executed by P1. Note that synchronization is needed for the counter c. When the *loop* is executed more than once, the algorithm requires local data to be generalized one step further before computing the next γ_c^1 at Step 5. Also, the equality test between ω_c^1 and ω_c^2 at step 8 is performed securely. This equality test can be implemented in various ways. The generic secure circuit evaluation approach can be used, or secure comparison protocols proposed in [6,8,16] can be applied. At step 9, the symbol $\bowtie$ represents the one-to-one join operator on the ID attribute (not part of the quasi-identifier, although it can be used in the join but dropped from the k-anonymous dataset as in [27]).

4.2 Proof of correctness

In this section, we prove Algorithm 4 achieves global k-anonymity. Referring to notations adopted in Sect. 4.1, let γ_c^1, γ_c^2 be synchronously computed from P1 and P2's locally k-anonymous data, and use the operators $\equiv, \neq$ defined in Definition 3. Define $T1_{\gamma_c^1}$ and $T2_{\gamma_c^2}$ as the locally k-anonymous data related to γ_c^1 and γ_c^2, respectively.

Theorem 5 *If $\gamma_c^1 \equiv \gamma_c^2$, then $T_k[QI] \leftarrow T1_{\gamma_c^1} \bowtie T2_{\gamma_c^2}$ satisfies global k-anonymity, where $T1_{\gamma_c^1} \bowtie T2_{\gamma_c^2}$ indicates creating global dataset through one-to-one join on two locally k-anonymous datasets.*

Proof We prove the above theorem by contrapositive. In other words, we prove the following statement: If $T_k[QI]$ does not satisfy global k-anonymity, then $\gamma_c^1 \neq \gamma_c^2$. Suppose $T_k[QI]$ is not k-anonymous, then there exists a subset of identical records $S = \{t_1, \ldots, t_j\} \subset T_k[QI]$ such that $0 < |S| < k$ or $0 < j < k$. Let $t_j[\gamma_c^1]$ denote the portion of the record t_j related to γ_c^1 stored at P1 and $t_j[\gamma_c^2]$ denote the portion of the record related to γ_c^2 stored at P2. Then, $\{t_1[\gamma_c^1], \ldots, t_j[\gamma_c^1]\}$ must be contained in some subset $\gamma_c^1[p]$, and $\{t_1[\gamma_c^2], \ldots, t_j[\gamma_c^2]\}$ must be contained in some subset $\gamma_c^2[q]$; as a result, $0 < |\gamma_c^1[p] \cap \gamma_c^2[q]| < k$. According to Definition 3, γ_c^1 and γ_c^2 are *not-anonymized* ($\gamma_c^1 \neq \gamma_c^2$). Thus, the contrapositive statement is true, so Theorem 5 holds. □

Theorem 6 *The executions of the SSI_t protocol on each pair $\theta_c^1[j], \theta_c^2[j]$ are sufficient; in other words, they cover all the necessary* anonymity *tests to justify that $\gamma_c^1 \equiv \gamma_c^2$.*

Proof Suppose the above statement is not true, then there must exist a pair $\gamma_c^1[p], \gamma_c^2[q]$, for some p, q, whose intersection size has not been compared with k and could potentially cause $\gamma_c^1 \neq \gamma_c^2$. In other words, this pair does not match any pair $\theta_c^1[j], \theta_c^2[j]$, where $1 \leq j \leq M$. If $|\gamma_c^1[p] \cap \gamma_c^2[q]| = 0$ and according to Definition 3, the pair is useless to justify whether or not γ_c^1 and γ_c^2 are *anonymized*. Otherwise, $\exists id \in \gamma_c^1[p] \cap \gamma_c^2[q]$. Based on the construction of θ_c^i, the pair $\gamma_c^1[p], \gamma_c^2[q]$ must match the pair $\theta_c^1[id], \theta_c^2[id]$. That means the size of $\gamma_c^1[p] \cap \gamma_c^2[q]$ was already examined against k before concluding whether or not $\gamma_c^1 \equiv \gamma_c^2$. This contradicts the assumption. □

Theorem 7 *$\gamma_c^1 \equiv \gamma_c^2$ if and only if $\omega_c^1 = \omega_c^2$.*

Proof $\Rightarrow$: If $\gamma_c^1 \equiv \gamma_c^2$, as described in Sect. 4.1, it holds that $|\theta_c^1[j] \cap \theta_c^2[j]| \geq k$. Then according to the definition of the SSI_t protocol, the execution of SSI_t on any pair $\theta_c^1[j], \theta_c^2[j]$ against k always returns 0 ($= b_1 \oplus b_2$). Based on the XOR operator, if the random bit generated by P2 is 1 during any execution of the SSI_t protocol, P1 must receive bit 1 during the same execution. On the other hand, if the random bit generated by P2 is 0, P1 must receive bit 0. Therefore, $\omega_c^1 = \omega_c^2$.

$\Leftarrow$: If $\omega_c^1 = \omega_c^2$, due to the characteristics of the XOR operator, the execution of SSI_t on every pair $\theta_c^1[j], \theta_c^2[j]$

Algorithm 5 DkA Simulator

Require: Private Data $T1$, Global k-anonymous dataset T_k, QI_{P1} (a subset of QI related to P1), Constraint k, $VGH_{A_i}, \forall A_i \in QI_{P1}$, assume $k \leq |T1| = M$

1: P1 generalizes $T1$ to be locally k-anonymous;
2: int $c \leftarrow 0$;
3: **repeat**
4: $c = c + 1$
5: P1 computes k-anonymous subsets $\hat{\gamma}_c^1$;
6: P1 computes $\hat{\theta}_c^1$, the "by identifier" subset list;
7: P1 computes $\hat{\omega}_c^1$ (by generating a random bit string with size M);
8: **until** $T1_{\hat{\gamma}_c^1} = T_k[QI_{P1}]$

must be 0. In other words, $|\theta_c^1[j] \cap \theta_c^2[j]| \geq k$ for $1 \leq j \leq M$. Based on Theorem 6 and Definition 3, every pair $\gamma_c^1[p], \gamma_c^2[q]$ needed to justify if γ_c^1 and γ_c^2 are *anonymized* has been considered, and none of the pairs violates the condition of being *anonymized*. Therefore, it must be true that $\gamma_c^1 \equiv \gamma_c^2$. □

We also need to note that the DkA framework adopts an iterative process, and its convergence depends on the local k-anonymization algorithm. As long as the local algorithm converges, the framework will eventually terminates due to the fact that each step causes a local change in the direction of convergence; eventually each local algorithm will converge leading to global termination.

4.3 Proof of security

Theorem 8 *Algorithm 4 is secure under the semi-honest definition of SMC.*

Proof To prove the framework is secure, we only need to show that the execution image of the framework can be simulated given the final outcome (the k-anonymous dataset). Algorithm 5 provides such a simulator for the framework DkA. Note that the simulators for both P1 and P2 are the same. The simulator is written from P1's point of view. The main point of the simulator is that the number of times P1 needs to generalize its private data locally can be found by simply comparing the current generalized local data with $T_k[QI_{P1}]$. If they are not equal, P1 knows that it needs to further generalize its local dataset.

Let Π_S be the view produced from the simulator, then according to Algorithm 5, $\Pi_S = (T1, k, QI_{P1}, c, \hat{\gamma}_c^1, \hat{\theta}_c^1, \hat{\omega}_c^1, \hat{e})$, where $\hat{e}$ is an implicit parameter in the simulator indicating the simulated equality test result between ω_c^1 and ω_c^2. Let Π_R be the view during the real execution of the DkA framework, then according to Algorithm 4, $\Pi_R = (T1, k, QI_{P1}, c, \gamma_c^1, \theta_c^1, \omega_c^1, e)$, where e is the real equality test result between ω_c^1 and ω_c^2. According to the comparison between $T1_{\hat{\gamma}_c^1}$ and $T_k[QI_{P1}]$, the simulator can simulate the exact equality test result between ω_c^1 and ω_c^2. As a result, $\hat{e} = e$, and it can be easily verified that $\hat{\gamma}_c^1 = \gamma_c^1$ and $\hat{\theta}_c^1 = \theta_c^1$. Since both $\hat{\omega}_c^1$ and ω_c^1 are strings of random bits with the same length, the distributions of $\hat{\omega}_c^1$ and ω_c^1 are computationally indistinguishable. Thus, the distributions of the two views Π_S, Π_R are computationally indistinguishable. Since we are able to simulate the result of SSI_t and the subroutine protocols are secure, the composition theorem of [12] enables us to conclude that the overall algorithm is secure. This concludes that the DkA framework is secure under the semi-honest model in the context of SMC. □

4.4 Complexity analysis

Since the number of times the global generalization (steps 4–7) needs to be executed depends on the structure of each value generalization hierarchy and the specific algorithm used to compute locally k-anonymous dataset, it is not possible to provide precise complexity analysis for overall execution of the DkA framework. On the other hand, we do give a precise complexity analysis for each round of global generalization.

Theorem 9 *The number of encryptions and communication complexity of each round of global generalization are bounded by $O(M^2)$ and $O(M^2 \log N)$ respectively.*

Proof Most encryption and communication for each round of global generalization occurs at step 7 of Algorithm 4, so the complexity of each global generalization is bounded by the complexity of step 7. At step 7, the two parties collaboratively execute the SSI_t protocol on each pair $\theta_c^1[j], \theta_c^2[j]$. Because the number of such pairs is M (the size of the dataset or the number of records in the dataset) and the domain size of every element in $\theta_c^i[j]$ is M, then according to Theorem 4, step 7 requires $\text{O}(M) \times M$ number of encryptions and $\text{O}(M \log N) \times M$ communication (in bits). Consequently, the number of encryptions and communication complexity of each global generalization step are bounded by $\text{O}(M^2)$ and $\text{O}(M^2 \log N)$. □

As stated previously, DkA is a framework and cannot be directly used to achieve global k-anonymization. However, the proof of correctness and the complexity analysis are applied to any secure distributed k-anonymization protocol that utilizes this framework. Even though the proof of security presented above is merely related to DkA, it provides a general road map for constructing a security proof of a concrete protocol.

Details of constructing such protocol using the framework are presented next.

5 A case study: DkA-Datafly

We now show how to use the DkA framework to transform the centralized Datafly into a secure two-party distributed k-anonymization protocol. We refer to this new protocol as DkA-Datafly. In order to have a *successful* transformation, the following three requirements should be met:

- *Secure*: DkA-Datafly is secure under the SMC definitions.
- *k-Anonymization*: DkA-Datafly accomplishes global k-anonymization.
- *Preciseness*: The results produced from DkA-Datafly are as close as possible to the centralized Datafly.

Datafly is an attribute-based generalization algorithm, so the modification to the DkA framework is minor. Steps 2–5 of the Datafly algorithm presented in Algorithm 1 implies a constraint: data tuples or records that are already k-anonymous will not be generalized further.[3] Thus, in order to satisfy the *preciseness* requirement, k-anonymous portion of the dataset should be removed before the next iteration of DkA-Datafly. We make this point clear in Sect. 5.1.

5.1 DkA-Datafly

Key steps of the DkA-Datafly protocol are highlighted in Algorithm 6. The structure of DkA-Datafly is the same as the framework DkA presented in Algorithm 4, except for steps 9–12 where two additional functions have been employed:

1. *Gen_Further*(T, u): this function takes a dataset T and returns T', where T' has been generalized further from T according to a specific generalization rule specified by u. Rule u is flexible in that it can be any generalization rule as long as it is agreed upon by both parties. Here is an example of rule u that we will adopt in Sect. 6: First find the attribute having the most number of distinct values. Then, generalize the most specific values related to that attribute one step up in their value generalization hierarchies. In order to satisfy the *preciseness* requirement, the rule u presented above employs a similar heuristic as the Datafly algorithm.
2. *Extract_Kanon*(T, χ_c): this function returns a subset S_k of T that is already globally k-anonymous and updates T by removing S_k from T. χ_c has the following properties:
   - It is a bit string resulting from the bit-wise XOR of two bit strings: ω_c^1, ω_c^2 (the same strings generated at step 7 of Algorithm 4).
   - It is used to identify the globally k-anonymous subset: $S_k = \bigcup_{j \wedge \chi_c[j]=0} \left(\theta_c^1[j] \cap \theta_c^2[j]\right)$, where $\theta_c^i[j]$ denotes the *jth* element in θ_c^i, and $\chi_c[j]$ denotes the value of its *jth* bit.[4]

Algorithm 6 DkA-Datafly (shown for P1; P2 identical.)

Require: Private Data $T1$, QI_{P1} (a subset of QI related to P1), Constraint k, $VGH_{A_i}, \forall A_i \in QI_{P1}$, assume $k \leq |T1| = M$, T_k^1 (output dataset of P1), Datafly, Generalization rule u

1: P1 generalizes $T1$ to be locally k-anonymous by using Datafly, denoted as $T1'$;
2: int $c \leftarrow 0$;
3: $T_k^1 \leftarrow \emptyset$;
4: **repeat**
5: $c = c + 1$;
6: P1 computes k-anonymous subsets γ_c^1;
7: P1 computes θ_c^1, the "by identifier" subset list;
8: P1 obtains the "by identifier" comparison shares ω_c^1 by collaboratively executing SSI_t once on each pair $\theta_c^1[j], \theta_c^2[j]$, for $1 \leq j \leq M$;
9: P1 and P2 collaboratively compute $\chi_c = \omega_c^1 \bigoplus \omega_c^2$
10: $T_k^1 \leftarrow T_k^1 \cup Extract\_Kanon(T1', \chi_c)$;
11: $T1' \leftarrow Gen\_Further(T1', u)$;
12: **until** $|T1'| < k$;
13: $return\ T_k[QI] \leftarrow T_k^1 \bowtie T_k^2$;

Next we present an example showing how S_k is computed from P1's point of view. Let $T1$ be the local dataset of P1 and $\theta_1^1, \theta_1^2, \omega_1^1, \omega_1^2$ be the same values computed in Sect. 4. P1, with the help of P2, gets $\chi_1 = \omega_1^1 \bigoplus \omega_1^2 = 111100001100$. Then $S_k = \bigcup_{j \wedge \chi_1[j]=0} \left(\theta_1^1[j] \cap \theta_1^2[j]\right) = \{5, 6, 7, 8, 11, 12\}$. The data related to S_k are removed before the next iteration of DkA-Datafly. (As we shall see, we can determine the data removed at each step from the final result.)

People may wonder if we could skip the computations of ω_c^i, χ_c and directly and securely compute the intersection between $\theta_1^1[j]$ and $\theta_1^2[j]$. In order for DkA-Datafly to guarantee the *security* requirement, the answer is negative since we should not learn the size of intersections smaller than k.

[3] For illustration purpose, the generalized data (Table 3) given in Sect. 4 did not follow this constraint. This does not affect any analysis regarding the DkA framework.

[4] Alternatively, we could compute $S_k = \bigcup_{\chi_c[j]=0} j$.

Algorithm 7 DkA-datafly Simulator

Require: Private Data $T1$, Global k-anonymous dataset T_k, QI_{P1} (a subset of QI related to P1), Constraint k, $VGH_{A_i}, \forall A_i \in QI_{P1}$, assume $k \leq |T1| = M$, $\hat{T}_k^1$, Datafly, Generalization rule u

1: P1 generalizes $T1$ to be locally k-anonymous by using Datafly, denoted as $\hat{T}1'$;
2: int $c \leftarrow 0$;
3: $\hat{T}_k^1 \leftarrow \emptyset$;
4: **repeat**
5: $c = c + 1$
6: P1 computes k-anonymous subsets $\hat{\gamma}_c^1$;
7: P1 computes $\hat{\theta}_c^1$, the "by identifier" subset list;
8: P1 randomly generates a bit string $\hat{\omega}_c^1$ with size M;
9: P1 compute $\hat{\chi}_c$ from $\hat{S}_k$;
10: P1 updates $\hat{T}_k^1$ based on $\hat{S}_k$;
11: P1 updates $\hat{T}1'$ based on $\hat{S}_k$ and u;
12: **until** $|\hat{T}1'| < k$;

The correctness of DkA-Datafly follows from the fact that during each iteration only globally k-anonymous data become part of the final output. This can be validated from the description of the *Extract_Kanon*(T, χ_c) function and the correctness proof of the DkA framework. Since the complexity of *Extract_Kanon*(T, χ_c) is less than the previous steps, the complexity of DkA-Datafly remains unchanged compared to DkA. Nevertheless, we need to show DkA-Datafly is secure.

5.2 Proof of security

Theorem 10 *Algorithm* 6 *is secure under the semi-honest definition of SMC.*

Proof The proof follows the same structure as the one presented for Algorithm 4. Thus, we need to design an algorithm that simulates the execution image of DkA-datafly. Algorithm 7 provides such a simulator. Note that the simulators for both P1 and P2 are the same. The simulator is written from P1's point of view.

Let I_{P1} be P1's input (i.e., $T1$, k, Datafly, u, etc). Let Π_R be the view during the real execution of DkA-Datafly, then according to Algorithm 6, $\Pi_R = (I_{P1}, c, \gamma_c^1, \theta_c^1, \omega_c^1, S_k, \chi_c^1, T_k^1, T1', e)$, where e is an implicit parameter in the protocol indicating the inequality test between the size of $T1'$ and k, and S_k is an implicit parameter representing the result computed from the *Extract_Kanon* function. Let Π_S be the view produced from the simulator, then according to Algorithm 7, $\Pi_S = (I_{P1}, c, \hat{\gamma}_c^1, \hat{\theta}_c^1, \hat{\omega}_c^1, \hat{S}_k, \hat{\chi}_c, \hat{T}_k^1, \hat{T}1', \hat{e})$. Compare the two views starting from the initial iteration:

- Before entering the *loop*, $T1' = \hat{T}1'$ because Datafly is a part of the input of P1.
- Then it follows: $\gamma_c^1 = \hat{\gamma}_c^1$ and $\theta_c^1 = \hat{\theta}_c^1$, where $c = 1$.
- Since $\hat{\omega}_c^1$ is randomly generated and based on the construction of ω_c^1, the two bit strings are computationally indistinguishable.
- S_k is a subset of $T1'$, and it is globally k-anonymous and a part of final output T_k, so S_k can be deduced from $T1'[QI_{P1}] \cap T_k[QI_{P1}]$. Similarly, $\hat{S}_k$ can be computed from $\hat{T}1'[QI_{P1}] \cap T_k[QI_{P1}]$. Because $T1' = \hat{T}1'$, according to the previous analysis, $S_k = \hat{S}_k$.

From the fact that $S_k = \hat{S}_k$, it can be easily verified that $\chi_c = \hat{\chi}_c, T_k^1 = \hat{T}_k^1, T1' = \hat{T}1'$ and $e = \hat{e}$. Because of these equalities, the indistinguishability between the two views does not changed from one iteration to another. Thus, the distributions of the two views Π_S, Π_R are computationally indistinguishable. This concludes that the DkA-Datafly protocol is secure under the semi-honest model in the context of SMC. □

Based on the previous analyses, DkA-Datafly satisfies the first two requirements: *security* and *k-anonymization*. Due to irregularities among data and value generalization hierarchies, it is very difficult to present a formal statement regarding the third requirement *preciseness*; however, we will validate it via comprehensive empirical results in Sect. 6.

6 Empirical analyses

Although the DkA-Datafly protocol adopts Datafly to produce locally k-anonymous dataset, the outcome of DkA-Datafly may not the same as that of Datafly when Datafly is used in a centralized environment. This is because each party performs a generalization at each step. For example, if Datafly would first generalize all the attributes held by P1, DkA-Datafly would instead generalize some attributes at P1 and the same number at P2, with the same method as Datafly used to independently choose which attributes at each site. This would presumably result in over-generalization.

To evaluate this effect, we compared DkA-Datafly with its centralized counterpart. In [30], the precision metric was used to evaluate the effectiveness of Datafly. Therefore, for consistency and unbiased evaluation, we utilize this metric to measure the quality of the outcome generated from DkA-Datafly. Before defining the metric, let make these notations clear:

- Let T denote a dataset, $QI = [A_1, \ldots, A_m]$ be the set of QI of T and VGH_{A_i} be the value generalization hierarchy (VGH) of attribute A_i.

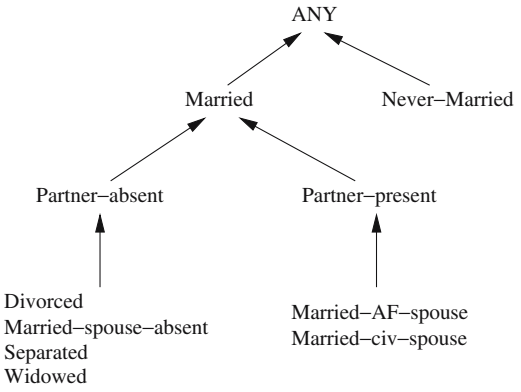


Fig. 3 Value generalization hierarchy for the marital status attribute

- T_k denotes the k-anonymous dataset with respect to T and QI. $T_k(i,j)$ denotes the value of A_i for the jth record in T_k, and $n = |T_k|$.
- Let $PL[T_k(i,j)]$ be the length of the path in VGH_{A_i} from the root to the leaf containing the value $T(i,j)$ that $T_k(i,j)$ was generalized from.
- Let $Height[T_k(i,j)]$ be the number of generalizations applied to get $T_k(i,j)$; i.e., the height in VGH_{A_i} above the most specific value $T(i,j)$.

For example, let VGH_{A_i} refer to the hierarchy presented in Fig. 3, and assume all values in the hierarchy are contained in T_k. Then, PL["Married"] = 3, PL ["Never-Married"] = 1, $Height$["Married"] = 2 and $Height$["Never-Married"] = 0. Formally, precision is defined as follows:

Definition 4 (Precision Metric [30])

$$PREC(T_k) = 1 - \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} Height[T_k(i,j)]}{\sum_{i=1}^{m}\sum_{j=1}^{n} PL[T_k(i,j)]}.$$

Informally, precision is a measure of distortion related to the original dataset. If the precision is 1, the values contained in the produced k-anonymous dataset are all original values. A precision of 0 indicates that each attribute is generalized to the most general value in its VGH.

6.1 Data description

We present results from k-anonymizing the *Adults* census dataset from the UC Irvine Machine Learning Repository [5]. Following the methodology of [17], we selected eight attributes which are good candidates for quasi-identifiers. These eight attributes have substantial variance among their VGHs and the number of distinct base values, giving the opportunity for significant variance in precision for different choices of which attributes to generalize. In particular, it leaves the possibility that the differences caused by DkA could reduce precision. Table 4 describes the attributes used. The dataset contains 30,162 records with no missing values. The VGHs that we adopted for these eight attribute are as used in [17]; a sample is shown in Fig. 3.

Table 4 Experimental dataset description

Attribute	Distinct values	VGH height
Education	16	4
Marital status	7	3
Native country	41	3
Occupation	14	2
Race	5	2
Relationship	6	2
Sex	2	1
Workclass	7	4

6.2 Experimental setup

The centralized Datafly provides a baseline precision. While DkA-Datafly is deterministic, the generalizations performed are dependent on the partitioning of the data between parties. To evaluate the potential loss of precision, we tried several different partitionings. We first varied the number of attributes assigned to each party, from a 1–7 split to an even 4–4 split. Within each split, we randomly selected five partitionings of data. We then ran the algorithm, and measured the precision. This was repeated for $k = 2$, 5, 10, 20, 50, and 100. In summary, we generate results as follows:

1. Compute $PREC_k$ (precision of centralized Datafly)
2. For each partition size $i = 1-4$, repeat the following steps five times:
   - Create two random vertical partitions: $T1, T2$ (i.e., i attributes are randomly assigned to $T1$, and the remaining attributes are assigned to $T2$)
   - Compute $PREC_{DkA}$ (precision of DkA-Datafly)
3. For each i across the five random executions, the maximum, minimum and average values of $PREC_{DkA}$ are recorded.

6.3 Experimental results

Figure 4 shows the results of these tests. The line represents the results of centralized Datafly (the line is simply shown for ease in identifying these results; it is not meant to indicate values between the points shown). For each value of k, we give the mean precision and high/low

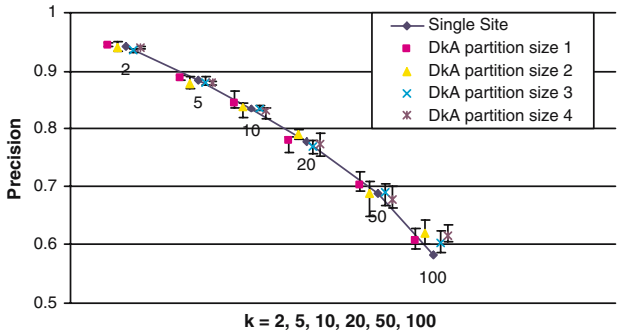


Fig. 4 Precision of DkA versus Datafly with various partitionings

values for each partition size. For example, the rightmost point (the dark $*$) gives the mean precision of 0.62 over five different partitionings, with four attributes going to each party, for $k = 100$. The error bars show that the precision over all five partitionings ranged from 0.60 to 0.63. Likewise, the light $\times$ gives results for partitions with three attributes going to one party and five to the other, etc.

For small k, the difference in precision between DkA-Datafly and Datafly is insignificant. As k increases, DkA-Datafly has the potential to give lower precision if the data partitioning happens to be particularly bad.[5] The results for $k = 100$ show an interesting anomaly; DkA-Datafly actually gives *better* precision than the centralized Datafly. We do not claim that DkA-Datafly can be expected to give better results; the goal is to approximate the results of the centralized algorithm. This result instead demonstrates that Datafly is a heuristic algorithm; this particular situation happens to be a bad case for the heuristic used. The partitioning forces the local Datafly instances to generalize differently, resulting in a better outcome.

The important result of these tests is that the variance in precision across different partitionings is small; the different choice of k is a much more significant factor. This validates that the decision to simultaneously generalize at both parties (enabling an efficient and fully secure algorithm) does not significantly impact the outcome; the resulting dataset is both guaranteed to be k-anonymous, and can be expected to have precision (and thus utility) comparable to a dataset anonymized by a central "trusted authority".

[5] Note that identifying and choosing a good partitioning of data is not interesting; in the real world the partitioning is governed by who holds which parts of the (private) data and is not subject to change.

6.4 Computational cost estimates

In this section, we estimate the practical cost of DkA-Datafly. The primary cost is encryption; by computing the number of encryptions required, we are able to estimate the total cost of the algorithm based on implementation of the encryption technique used. The complexity of DkA-Datafly is bounded by the number of times the SSI_t protocol is called; we first estimate the cost of SSI_t. Step 1 of Algorithm 2 determines the cost of SSI_t, and we base our estimation on this step only because when M (the size of the dataset) is large, the cost of the rest of the protocol is insignificant.

The values that need to be encrypted are either 0 or 1; therefore, the simulation adopts the homomorphic encryption scheme proposed in [24] (efficient for 0 or 1 encryptions). The scheme has the following parameters:

- *Public-Key*: (n, g, h, k), where $n = p^2q$, $g \in (\mathbb{Z}/n\mathbb{Z})$ is randomly chosen such that the order of $g^{p-1} \mod p^2$ is p, and $h = g^n \mod n$.
- *Secret-Key*: (p, q), where $|p| = |q| = k$.
- *Encryption*: $C = g^m h^r \mod n$, where m is the plaintext (0 or 1 in our problem domain) and $r \in (\mathbb{Z}/n\mathbb{Z})$ is randomly chosen.

We use the same parameters as those adopted in [24], especially the size of $n = 1,024$ bits (commonly used in public key encryption, and r is randomly chosen from $1, \ldots, 2^{130}$). Basically, step 1 of Algorithm 2 requires M homomorphic encryptions. As a result, we present the cost of SSI_t in terms of the number of seconds to perform M encryptions:

- Estimated cost of SSI_t: 19.5 s on average when $M =$ 30,162 (the size of our dataset). In other words, 19.5 s to perform 30,162 homomorphic encryptions.
- Platform: Intel®Xeon™ 3 GHz processor, with 1 GB RAM, hyper-threading enabled, and running the Linux SMP kernel version 2.6.14.6.

DkA-Datafly utilizes an iterative process, the time complexity of each iteration of DkA-Datafly is determined by the number of calls to the SSI_t protocol, and it is consequently determined by the number of encryptions. Let λ_1 be the number of the homomorphic encryptions needed for the first iteration of DkA-Datafly.[6] The first iteration requires M (30,162) number of calls of SSI_t; therefore, $\lambda_1 = 30,162 \times 30,162$. Since globally k-anonymous data are removed after each iteration, the

[6] Note that our estimates are from P1's point of view since P1 performs more computations than P2.

Table 5 Complexity of DkA-Datafly in terms of λ_1 (the first round cost)

k	20	50	100
Max.	$1.75 \times \lambda_1$	$2.35 \times \lambda_1$	$2.89 \times \lambda_1$
Min.	$1.07 \times \lambda_1$	$1.11 \times \lambda_1$	$1.11 \times \lambda_1$
Avg.	$1.53 \times \lambda_1 \approx 10.14$ days	$1.89 \times \lambda_1 \approx 12.53$ days	$2.43 \times \lambda_1 \approx 16.11$ days

size of following rounds are generally much smaller. Although dependent on the value of k, the size of the dataset remained for the second iteration is generally less than half that of the first iteration; in other words, the time to perform the homomorphic encryptions is generally less than 1/4 of the first round. We estimated that when $k = 100$, the total expected number of encryptions by DkA-Datafly is about 2.43 times of the number of encryptions required for the first iteration. Table 5 includes estimates for different k values, and it records the maximum, minimum and expected costs across different partition sizes and trials.

While these times may seem long, it is important to view them in the context in which DkA-Datafly would be used. When privacy concerns prevent sharing of data, the time required to construct a k-anonymous dataset from distributed sources is infinite. More practically, such release of data would require permission from all of the subjects involved, or at the very least, an involved approval process (e.g., Institutional Review Board approval for Human Subjects Research). Such approval, from each institution involved, is likely to take months. Given the guarantees of privacy provided by DkA-Datafly, fast review (such as Exempt status under U.S. Human Subjects Research regulations) may be possible, saving much more than the 2 weeks of (relatively inexpensive) CPU time to generate the dataset.

There are two simple methods to improve the efficiency of DkA-Datafly (or the DkA framework in general): *pre-computation* and *parallelization*. Since we know the values need to be encrypted are either 0 or 1, we can pre-compute and store encryptions of 0s and 1s. Then encryption becomes fetching an element from the corresponding set. In addition, step 8 of Algorithm 6 (DkA-Datafly) or step 7 of Algorithm 4 (DkA framework) is parallelizable because each pair of $\theta_c^1[j], \theta_c^2[j]$ is independent of the other.

7 Conclusion/future work

The knowledge in data has value; the more complete the data, the greater the potential knowledge. On the other hand, privacy of information in databases is an increasingly visible issue. Partitioning data, so as to prevent complete information from being available, is effective at preventing misuse of data, but it also makes beneficial use more difficult. One way to preserve privacy while enabling beneficial use of data is to utilize k-anonymity for publishing data. Maintaining the benefits of partitioning while generating integrated k-anonymous data requires a protocol that does not violate privacy through disclosure to the data holders.

In this paper, we have laid out this problem and presented a two-party framework DkA that is proven to generate a k-anonymous dataset while satisfying the security definition of SMC. DkA is very general in a sense that any centralized k-anonymization protocol can be used to compute locally k-anonymous data, and its structure can be effectively adopted to create a secure two-party k-anonymization protocol from an insecure centralized k-anonymization algorithm. DkA-Datafly provides such an application. In addition to being privacy-preserving, our empirical analyses have validated that the k-anonymous dataset computed by DkA-Datafly has comparable precision to that computed by Datafly.

While this framework is a general solution, there are still interesting issues to address with respect to the creation of k-anonymous datasets from distributed sources.

Based on the structure of DkA and the effectiveness of DkA-Datafly, we hypothesize that centralized k-anonymization algorithms (especially cell-level or attribute level based algorithms such as [15,28,30]) utilizing the sub-optimal structure stated in Theorem 1 can be easily and effectively transformed into a secure two-party protocol by using the DkA framework. Although the DkA framework can also be applied to full-domain generalization based algorithms (such as [3,19]), the effectiveness of such transformation needs to be validated through extensive empirical analyses.

There are generally two kind of approaches to compute a k-anonymous dataset: *bottom-up* and *top-down*. Datafly is a *bottom-up* algorithm in that it computes k-anonymous data by substituting a specific value with a more generalized value. A *top-down* k-anonymization algorithm does the opposite. If a k-anonymization protocol used to compute locally k-anonymous data adopts *top-down* strategy, the DkA framework can be modified slightly to produce the correct result as follows: the *loop* from steps 3–8 of Algorithm 4 should terminate whenever $\omega_c^1 \neq \omega_c^2$, and the join of locally k-anonymous

datasets from the round right before the termination round creates a globally k-anonymous dataset.

Another issue is how to extend DkA to a multiparty framework. While executing DkA between two parties, then joining the resulting dataset with a third, etc. would ensure k-anonymity, keeping the global identifier hidden during the process demands some thought. Another idea to explore is iteratively pairwise executing DkA among parties. Even though this approach would generate a globally k-anonymous dataset, it could violate security under the consideration of SMC.

Could we efficiently extend DkA to satisfy the security definition of a malicious adversary? It would seem that since DkA requires the local dataset to be k-anonymous before creating globally k-anonymous dataset, the most a malicious party can get is the other party's locally k-anonymous data. However, this can give the malicious party a global dataset that is less anonymous than joining their own dataset with a globally k-anonymous dataset, violating a strict standard of k-anonymity.

Privacy is a challenging area. Privacy does not yet have solid definitions corresponding to semantic security – it is instead a tradeoff between utility of data and the potential for misuse. We have presented a framework for generating a k-anonymous dataset that provides the utility of k-anonymity and ensures that the process of constructing the dataset does not pose any additional risk to privacy.

Acknowledgments We wish to thank Professor Elisa Bertino for comments and discussions that lead to this work.

Appendix

Here, we present a pure circuit based implementation of the SSI_t protocol whose key structure is highlighted in Algorithm 8. Refer to Algorithm 8, the *SUM* indicates a summation circuit computing the sum of M bits, and the *COMPARE* indicates a comparison circuit producing bit 1 if $d < t$.

Algorithm 8 Circuit-based SSI_t Protocol

Require: $(s_1, t), (s_2, t, b_2), |s_1| = |s_2| = M$
1: $c[i] = s_1[i] \wedge s_2[i]$, for $1 \leq i \leq M$
2: $d = SUM(c[1], c[2], \ldots, c[M])$
3: $b_1 = COMPARE(d, t) \oplus b_2$
4: return $(b_1, \perp)$

Complexity of circuit-based SSI_t

Since the size of s_j is M, the number of AND gates for step 1 in Algorithm 8 is M. Because each $c[i]$ is a single bit, the number of gates of the summation circuit (or the number of gates required at step 2 in Algorithm 8) is bounded by $O(M)$. The size of the sum resulted at step 2 of Algorithm 8 is no bigger than $\log M$ number of bits, so the comparison circuit at step 3 is bounded by $O(\log M)$. According to the above analysis, the number of gates of SSI_t is bounded by $O(M)$.

In general, the number of oblivious transfers (OT) required for secure circuit evaluation is bounded linearly on the size of the circuit.[7] Suppose the range of the key adopted in a OT protocol is bounded by N, the size of the key is bounded by $\log N$. As a consequence, the communication complexity of SSI_t is $O(M \log N)$ in bits. Note that although this circuit-based SSI_t protocol has the same complexity as SSI_t presented in Sect. 3, the constant cost of OT is expected to be higher (in number of encryptions) than the direct encryption used in SSI_t.

References

1. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of ACM SIGMOD international conference on Management of Data. San Diego, California, 9–12 June 2003. http://doi.acm.org/10.1145/872757.872771
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD Conference on Management of Data. pp. 439–450. ACM Dallas, TX, 14–19 May 2000, http://doi.acm.org/10.1145/342009.335438
3. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proceedings of the IEEE International Conference on Data Engineering. Tokyo, Japan, 5–8 2005 April
4. Benaloh, J.C.: Secret sharing homomorphisms: Keeping shares of a secret. In: Advances in Cryptography, CRYPTO86: Proceedings, Odlyzko, A. (ed.) vol. 263. pp. 251–260. Springer-Verlag, Lecture Notes in Computer Science, 1986, http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9 743&volume=263&spage=251
5. Blake, C., Merz, C.: UCI repository of machine learning databases. 1998. http://www.ics.uci.edu/∼mlearn/ MLRepository.html
6. Boudot, F., Schoenmakers, B., Traore, J.: "A fair and efficient solution to the socialist millionaires' problem. Discrete Appl. Math. **111** (1-2), 23–36 2001. http://www.win.tue.nl/∼berry/papers/dam.pdf
7. Dobkin, D., Jones, A.K., Lipton, R.J.: Secure databases: Protection against user influence. ACM Trans. Database Syst **4**(1), 97–106 1979. http://doi.acm.org/10.1145/320064.320068
8. Fischlin, M.: A cost-effective pay-per-multiplication comparison method for millionaires. RSA Secu 2001 Cryptographer's Track **2020**(1–2), 457–471, 2001. http://www.mi.informatik.uni-frankfurt.de/research/papers/fischlin.millionaire.2001.pdf
9. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Eurocrypt 2004. Interna-

[7] Details regarding secure circuit evaluation can be found in [11] and [12], and the concept of oblivious transfer can be found in [23] and [25].

tional Association for Cryptologic Research (IACR), Interlaken, Switzerland: 2–6 2004 May
10. Goethals, B., Laur, S., Lipmaa, H., Mielikainen, T.: On secure scalar product computation for privacy-preserving data mining. In: Park, C., Chee, S., (eds.), The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004), Seoul, Korea, 2–3 2004 December
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game - a completeness theorem for protocols with honest majority. In: 19th ACM Symposium on the Theory of Computing, 1987, pp. 218–229. http://doi.acm.org/10.1145/28395.28420
12. Goldreich, O.: The Foundations of Cryptography.Cambridge University Press, 2004, vol. 2, ch. General Cryptographic Protocols. http://www.wisdom.weizmann.ac.il/∼oded/ PSBookFrag/prot.ps
13. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
14. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC'85), pp. 291–304 Providence, Rhode Island, USA, 6–8 1985 May
15. Hundepool, A., Willenborg, L.: μ- and τ-argus: software for statistical disclosure control. In: Third International Seminar on Statistical Confidentiality (1996)
16. Ioannidis, I., Grama, A.: An efficient protocol for yao's millionaires' problem. In: Hawaii International Conference on System Sciences (HICSS-36), pp. 205–210. 2003 Waikoloa Village, Hawaii, 6–9 2003 January
17. Iyengar, V.S.: Transforming data to satisfy privacy constraints, In: Proceedings of the 2002 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 23–26 Edmonton, Alberta, Canada, 2002
18. Jiang, W., Clifton, C.: Privacy-preserving distributed *k*-anonymity. In: Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Database and Applications Security, Storrs, Connecticut, 7–10 2005 August
19. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, MD, 13–16 2005 June
20. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004).: ACM Press, Paris, France 14–16 2004 June
21. Moore, R.A., Jr., Controlled data-swapping techniques for masking public use microdata sets. U.S. Bureau of the Census, Washington, DC., Statistical Research Division Report Series RR 96-04, 1996. http://www.census.gov/srd/papers/pdf/rr96-4.pdf
22. Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: Proceedings of the 5th ACM conference on Computer and communications security. pp. 59–66. ACM Press, San Francisco, California, United States (1998)
23. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation In: Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing. pp. 245–254. ACM Press, Atlanta, Georgia, United States: (1999)
24. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In:Advances in Cryptology-Eurocrypt'98, LNCS 1403. pp. 308–318. Springer, Berlin Heidelberg Newyork(1998)
25. Rabin, M.: How to exchange secrets by oblivious transfer. Aiken Computation Laboratory, Harvard University, Tech. Rep. TR-81, 1981
26. Samarati, P.: Protecting respondent's privacy in microdata release. IEEE Trans. Knowl. Data. Eng. **13**(6), 1010–1027 (2001)
27. Schadow, G., Grannis, S.J., McDonald, C.J.: Privacy-preserving distributed queries for a clinical case research network. In: Clifton, C., Estivill-Castro, V.: (eds), IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining, vol. 14. pp. 55–65. Australian Computer Society, Maebashi City, Japan: 9 2002 December, http://crpit.com/Vol14.html
28. Sweeney, L.: Guaranteeing anonymity when sharing medical data, the datafly system. Proc. J. Am. Med. Inform. Assoc. (1997)
29. Sweeney, L.: Computational disclosure control: A primer on data privacy protection. Ph.D. dissertation, Massachusetts Institute of Technology, (2001)
30. Sweeney, L.: Achieving *k*-anonymity privacy protection using generalization and suppression. Int J Uncertainty, Fuzziness and Knowledge-based Syst **10**(5), 571–588, 2002. http://privacy.cs.cmu.edu/dataprivacy/ projects/kanonymity/kanonymity2.html
31. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertainty, Fuzziness and Knowledge-based Syst. **10**(5), 557–570 2002. http://privacy.cs.cmu.edu/dataprivacy/projects/kanonymity/kanonymity.html
32. Vaidya, J.: Privacy preserving data mining over vertically partitioned data. Ph.D. dissertation, Purdue University, West Lafayette, Indiana, 2004. http://www.cs.purdue.edu/homes/jsvaidya/thesis.pdf
33. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. J. Comput. Sec. **13**(4), 593–622 (2005)
34. Wright, R.N., Yang, Z.: Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In: Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, 22–25 2004 August
35. Yao, A.C.: Protocols for secure computation. In: Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science. pp. 160–164. IEEE, 1982
36. Yao, A.C.: How to generate and exchange secrets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science. pp. 162–167. IEEE, 1986
37. Zhong, S., Yang, Z., Wright, R.N.: Privacy-enhancing *k*-anonymization of customer data. In: Proceedings of the 24rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2005). ACM Press, Baltimore, Maryland, 13–16 2005 June