Privacy-Enhancing k-Anonymization of Customer Data *

Sheng Zhong[1,2] Zhiqiang Yang[1] Rebecca N. Wright[1]
[1] Computer Science Department, Stevens Institute of Technology, Hoboken, NJ 07030, USA
[2] DIMACS Center, Rutgers University, Piscataway, NJ 08854, USA
{sz38|zyang|rwright}@cs.stevens.edu

ABSTRACT

In order to protect individuals' privacy, the technique of k-anonymization has been proposed to de-associate sensitive attributes from the corresponding identifiers. In this paper, we provide privacy-enhancing methods for creating k-anonymous tables in a distributed scenario. Specifically, we consider a setting in which there is a set of customers, each of whom has a row of a table, and a miner, who wants to mine the entire table. Our objective is to design protocols that allow the miner to obtain a k-anonymous table representing the customer data, in such a way that does not reveal any extra information that can be used to link sensitive attributes to corresponding identifiers, and without requiring a central authority who has access to all the original data. We give two different formulations of this problem, with provably private solutions. Our solutions enhance the privacy of k-anonymization in the distributed scenario by maintaining end-to-end privacy from the original customer data to the final k-anonymous results.

1. INTRODUCTION

In today's information society, given the unprecedented ease of finding and accessing information, protection of privacy has become a very important concern. In particular, large databases that include sensitive information (e.g., health information) have often been available to public access, frequently with identifiers stripped off in an attempt to protect privacy. However, if such information can be associated with the corresponding people's identifiers, perhaps using other publicly available databases, then privacy can be seriously violated. For example, Sweeney [32] pointed out that one can find out who has what disease using a public database and voter lists. To solve such problems, Samarati and Sweeney [27] have proposed a technique called *k-anonymization*. In this paper, we study how to enhance privacy in carrying out the process of k-anonymization.

*This work was supported by the National Science Foundation under grant number CCR-0331584.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PODS 2005 June 13-15, 2005, Baltimore, Maryland.
Copyright 2005 ACM 1-59593-062-0/05/06 . . . $5.00.

Consider a table that provides health information of patients for medical studies, as shown in Table 1. Each row of the table consists of a patient's date of birth, zip code, allergy, and history of illness. Although the identifier of each patient does not explicitly appear in this table, a dedicated adversary may be able to derive the identifiers of some patients using the combinations of date of birth and zip code. For example, he may be able to find that his roommate is the patient of the first row, who has allergy to penicillin and a history of pharyngitis.

Date of Birth	Zip Code	Allergy	History of Illness
03-24-79	07030	Penicillin	Pharyngitis
08-02-57	07028	No Allergy	Stroke
11-12-39	07030	No Allergy	Polio
08-02-57	07029	Sulfur	Diphtheria
08-01-40	07030	No Allergy	Colitis

Table 1: A Table of Health Data

In this example, the set of attributes {date of birth, zip code} is called a *quasi-identifier* [12, 32], because these attributes in combination can be used to identify an individual with a significant probability. In this paper, we say an attribute is a *quasi-identifier attribute* if it is in the quasi-identifier. The attributes like allergy and history of illness are called *sensitive attributes*. (There may be other attributes in a table besides the quasi-identifier attributes and the sensitive attributes; we ignore them in this paper since they are not relevant to our investigation.) The privacy threat we consider here is that an adversary may be able to link the sensitive attributes of some rows to the corresponding identifiers using the information provided in the quasi-identifiers. A proposed strategy to solve this problem is to make the table *k-anonymous* [27].

Date of Birth	Zip Code	Allergy	History of Illness
*	07030	Penicillin	Pharyngitis
08-02-57	0702*	No Allergy	Stroke
*	07030	No Allergy	Polio
08-02-57	0702*	Sulfur	Diphtheria
*	07030	No Allergy	Colitis

Table 2: 2-Anonymized Table of Health Data

In a k-anonymous table, each value of the quasi-identifier appears at least k times. Therefore, if the adversary only uses

the quasi-identifiers to link sensitive attributes to the identifiers, then each involved entity (patient in our example) is "hidden" in at least k peers. The procedure of making a table k-anonymous is calld *k-anonymization*. It can be achieved by *suppression* (i.e., replacing some entries with "$*$") or *generalization* (e.g., replacing some or all occurrences of "07028" and "07029" with "0702$*$"). Table 2 shows the result of 2-anonymization on Table 1.

Several algorithmic methods have been proposed describing how a central authority can k-anonymize a table before it is released to the public (e.g. [30, 27, 26, 32, 31, 24, 8]). In this paper, we consider a related but different scenario: distributed customers holding their own data interact with a miner and use k-anonymization in this process to protect their own privacy. For example, imagine the above mentioned health data are collected from customers by a medical researcher. The customers will feel more comfortable if the medical researcher does not need to be trusted and only sees a k-anonymized version of their data. To achieve this, we show methods by which k-anonymization can be jointly performed by the involved parties in a private manner such that no single participant, including the miner, learns extra information that could be used to link sensitive attributes to corresponding identifiers.

1.1 Our Contributions

We give privacy-enhancing methods for creating k-anonymous tables in a distributed scenario. Our methods do not require a central authority who has access to all the original data, nor do they require customers to share their data with each other.

Specifically, we consider a setting in which there is a set of customers, each of whom has a row of a table, and a miner, who wants to mine the entire table. Our objective is to design protocols that allow the miner to obtain a k-anonymous table representing the customer data in such a way that does not reveal any extra information that can be used to link sensitive attributes to corresponding identifiers. We give two different formulations of this problem:

- In the first formulation, given a table, the protocol needs to extract the *k-anonymous part* (i.e., the maximum subset of rows that is already k-anonymous) from it. The privacy requirement is that the sensitive attributes outside the k-anonymous part should be hidden from any individual participant including the miner. This formulation is suitable if the original table is already close to k-anonymous.
- In the second formulation, given a table, the protocol needs to suppress some entries of the quasi-identifier attributes, so that the entire table is k-anonymized. The privacy requirement is that the suppressed entries should be hidden from any individual participant. This formulation is suitable even if the original table is not close to k-anonymous.

We present efficient solutions to both problem formulations. Our solutions use cryptography to obtain provable guarantees of their privacy properties, relative to standard cryptographic assumptions. Our solution to the first problem formulation does not reveal any information about the sensitive attributes outside the k-anonymous part. Our solution to the second problem formulation is not fully private, in that it reveals the k-anonymous result as well as the distances between each pair of rows in the original table. We prove that it does not reveal any additional information. Our protocols enhance the privacy of k-anonymization by maintaining end-to-end privacy from the original customer data to the final k-anonymous results.

We briefly overview related work in Section 2. In Section 3, we formalize our two problem formulations. Our solutions are presented in Sections 4 and 5, respectively. We conclude in Section 6.

2. RELATED WORK

As a strategy to prevent identity disclosure in microdata release, k-anonymization was first proposed and analyzed by Samarati and Sweeney [30, 27, 26, 32, 31]. Meyerson and Williams [24] formally studied how to minimize the number of suppressed entries in k-anonymization and showed that it is NP-hard; they then gave approximation algorithms for this problem. Aggarwal et al. [4] showed that the problem is NP-hard even if the attributes are ternary-valued; they also gave algorithms with improved approximation ratios. Bayardo and Agrawal [8] studied optimal k-anonymization with more cost metrics and proposed a practical solution.

The research area of statistical databases has studied how to protect individual privacy while supporting information sharing. There is a rich literature on privacy in statistical databases; interested readers can refer to surveys [2, 29]. Proposed methods can be categorized into query restriction (e.g., [22, 11]) and data perturbation (e.g., [25, 33, 9, 1]). In particular, the tradeoff between privacy and utility in statistical databases was investigated by Dinur and Nissim [14].

Another related area is privacy-preserving data mining, which also considers protection of sensitive data while maintaining data utility. Representative work in this area includes, among others, [6, 5, 17, 16, 21, 23, 34, 35, 20]. In addition, Aggarwal and Yu propose an approach called condensation [3] to produce publishable data that protects privacy yet provides utility for data mining applications. Their approach condenses data in groups, where the minimum size of a group is predetermined. The records in a group are all randomized such that only a few statistic properties of these records are kept.

Privacy is studied extensively in various aspects of cryptography. General results on cryptographic protocols are summarized in [19]. However, as mentioned in [19], the constructions presented in the proofs of these results cannot be directly applied in general, particularly for large amounts of input data (in our case, a large number of customers), because they are prohibitively expensive.

3. PROBLEM FORMULATIONS

Consider a table with m quasi-identifier attributes, ($s_1, \ldots,$ s_m), and n sensitive attributes, $(a_1, \ldots, a_n)$. Without loss of generality, we assume that there are no other attributes except these $m+n$. Suppose that there are $N+1$ involved parties: N customers and one miner, and that all these parties are polynomial-time bounded. For convenience, the miner assigns indices 1 through N to the customers; in the sequel, by "customer i" we mean "the customer with index i". Note that the indices are *not* identifiers because they are arbitrar-

ily assigned by the miner, who does not know the identifiers of the customers. Each customer i has a row of the table, which is denoted by $R_i = (s_1^{(i)}, \ldots, s_m^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)})$.

We assume there are private unidentified channels between each customer and the miner. That is, the channels are untappable and the miner has no information about which customer is using which channel, but each channel is used by exactly one customer. (These properties can be provided using standard cryptographic techniques.)

We rigorously define our privacy requirement by adapting the standard definition of privacy [19] for cryptographic protocols in the *semi-honest model* to our setting. In the semi-honest model, each party is assumed to follow the protocol but parties may attempt to derive extra information to violate privacy of other parties. This model has been extensively studied in cryptography (cf. [19]) and widely applied to privacy problems with large-size data (e.g., [23, 20]). Although the semi-honest model places a strong restriction on participants' behavior, there are at least two reasons for studying our problem in this model. First, deviating from the protocol requires a considerable amount of effort (to hack the computer program). In an application like collecting health data from customers, it may be reasonable to assume that the participants are not willing or able to invest that amount of effort on violating others' privacy. Second, it has been shown that any protocol private in the semi-honest model can be "translated" to one secure in the fully malicious model in which parties may deviate arbitrarily from their specified protocols [19], though at a substantial increase in the cost of the solution. If we modify this translated protocol to improve efficiency, we may be able to obtain a practical solution in the malicious model as well.

In defining our privacy requirement, we assume that before the protocol starts, there exists a global private key for a public key cryptosystem[1], which is shared among the customers in the sense of *secret sharing* [28]. Each customer is "preloaded" with up to a constant number of shares. Together, the shares form the global private key: Each share is only known to its owner; no customer knows the actual global key. (Such a situation can be established without a central authority by a *distributed key generation* protocol such as [18].)

Our overall objective is to enable the miner to obtain a k-anonymized table in a private manner (so that he can mine the table). As mentioned in Section 1, this can be achieved in two ways, described in detail in Sections 3.1 and 3.2: either we enable the miner to extract the k-anonymous part of the table, or we enable him to obtain a k-anonymized table in which some entries of the quasi-identifier attributes are suppressed.

3.1 Formulation 1: Private Extraction of k-Anonymous Part

In the first problem formulation, the miner extracts the k-anonymous part of the table (i.e., the maximum subset of rows that is k-anonymous), but does not learn extra information about the sensitive attributes of the rows outside the k-anonymous part. Consequently, the miner cannot link the sensitive attributes of any row to the corresponding identifiers.

Intuitively, our privacy requirement states that, for each party (miner or customer), the view of the protocol seen by that party can be simulated by an algorithm that has no knowledge of the sensitive attributes outside the k-anonymous part. This captures the requirement that any individual party cannot learn any extra information about these sensitive attributes by virtue of engaging in the protocol.

To formalize this requirement, we must first define the *view* of each party: during an execution of the protocol, a party's view consists of this party's data and preloaded key shares (if any), all the coin flips of this party, and all the messages this party receives. We denote by $\mathsf{view}_{\mathsf{miner}}(T)$ ($\mathsf{view}_i(T)$, resp.) the view of the miner (customer i, resp.) during an execution with the table

$$\begin{aligned} T &\stackrel{\mathsf{def}}{=} \{R_i : i \in [1, N]\} \\ &= \{(s_1^{(i)}, \ldots, s_m^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)}) : i \in [1, N]\}. \end{aligned}$$

In the sequel, we denote by $\mathcal{K}(T)$ the k-anonymous part of the table T. The notation $\stackrel{\mathsf{c}}{\equiv}$ denotes *computational indistinguishability* of *probability ensembles*. Readers can refer to, e.g., [19], for the definitions of probability ensembles and computational indistinguishability.

DEFINITION 1. *A protocol for extracting $\mathcal{K}(T)$ is* ideally private *if there exist $N+1$ probabilistic polynomial-time algorithms M, $M_1, \ldots, M_N$ such that*

$$\{M(\mathsf{keys}_{\mathsf{miner}}, \mathcal{K}(T), \{(s_1^{(i)}, \ldots, s_m^{(i)}) : i \in [1, N]\})\}_T \stackrel{\mathsf{c}}{\equiv} \{\mathsf{view}_{\mathsf{miner}}(T)\}_T,$$

and that, for any $i \in [1, N]$,

$$\{M_i(\mathsf{keys}_i, R_i, \mathcal{K}(T), \{(s_1^{(i)}, \ldots, s_m^{(i)}) : i \in [1, N]\})\}_T \stackrel{\mathsf{c}}{\equiv} \{\mathsf{view}_i(T)\}_T,$$

where $\mathsf{keys}_{\mathsf{miner}}$ (keys_i, resp.) denotes the miner's (customer i's, resp.) preloaded key shares (if any). The algorithms M and M_i for $i \in [1, N]$ are called simulators *(for the miner and customer i, respectively).*

3.2 Formulation 2: k-Anonymization by Privately Suppressing Entries

One method for k-anonymizing a table is to suppress entries—ideally suppressing as few as possible [24, 4]. Our second problem formulation supports suppression in our distributed setting. Let $\mathsf{Anonymized}(T)$ denote the output (which is a k-anonymized table) of a protocol that k-anonymizes the table T by suppressing entries. We have an analogous privacy requirement in this case as in Formulation 1, except that the privacy is relative to $\mathsf{Anonymized}(T)$ instead of $\mathcal{K}(T)$ and the quasi-identifier:

DEFINITION 2. *A protocol for k-anonymization by suppressing entries is* ideally private *if there exist $N+1$ probabilistic polynomial-time algorithms (called* simulators*) M, $M_1, \ldots, M_N$ such that*

$$\{M(\mathsf{keys}_{\mathsf{miner}}, \mathsf{Anonymized}(T))\}_T \stackrel{\mathsf{c}}{\equiv} \{\mathsf{view}_{\mathsf{miner}}(T)\}_T,$$

[1]Throughout this paper, by "key" we mean a cryptographic key. To avoid confusion, we do *not* use the term "key" in the sense of a database key attribute.

and that, for any $i \in [1, N]$,

$$\{M_i(\mathsf{keys}_i, R_i, \mathsf{Anonymized}(T))\}_T \stackrel{\mathsf{c}}{\equiv} \{\mathsf{view}_i(T)\}_T,$$

where $\mathsf{keys}_{\mathsf{miner}}$ *(*keys_i*, resp.) denotes the miner's (customer* i*'s, resp.) preloaded key shares (if any).*

In our solution for Formulation 2, we are unable to satisfy the ideal privacy of Definition 2. (General cryptographic solutions exist that could provide ideal privacy, but at much greater computation and communication costs.) Instead, we achieve a relaxed, but well-defined, notion of privacy in which a specified (and presumably small) amount of information is revealed. Formally:

DEFINITION 3. *Let* $\mathcal{F}(T)$ *be a function of the table* T*. A protocol for* k*-anonymization by suppressing entries* leaks only $\mathcal{F}(T)$ *if there exist probabilistic polynomial-time algorithms (called* simulators*)* M *and* $M_1, \ldots, M_N$ *such that*

$$\{M(\mathsf{keys}_{\mathsf{miner}}, \mathsf{Anonymized}(T), \mathcal{F}(T))\}_T \stackrel{\mathsf{c}}{\equiv} \{\mathsf{view}_{\mathsf{miner}}(T)\}_T,$$

and that, for any $i \in [1, N]$,

$$\{M_i(\mathsf{keys}_i, R_i, \mathsf{Anonymized}(T), \mathcal{F}(T))\}_T \stackrel{\mathsf{c}}{\equiv} \{\mathsf{view}_i(T)\}_T,$$

where $\mathsf{keys}_{\mathsf{miner}}$ *(*keys_i*, resp.) denotes the miner's (customer* i*'s, resp.) preloaded key shares (if any).*

4. OUR SOLUTION FOR FORMULATION 1

In this section, we solve the first formulation of the problem. That is, we design a protocol that privately extracts the k-anonymous part of a table. The basic idea of our design is that each customer encrypts her sensitive attributes using an encryption key that can be derived if and only if there are at least k rows whose quasi-identifiers are equal. Specifically, the key to encrypt the sensitive attributes $(a_1^{(i)}, \ldots, a_n^{(i)})$ is a function of the corresponding quasi-identifier $(s_1^{(i)}, \ldots, s_m^{(i)})$ and it is shared among the customers with threshold k in the sense of secret sharing (see below for explanation of secret sharing). Each customer submits to the miner one share of the key(s) corresponding to her quasi-identifier. As a result, if and only if there are at least k customers whose quasi-identifiers are equal, the miner is able to recover the appropriate decryption key.

The remaining technical question is how each customer selects the key. On the one hand, we do not want every customer with the same quasi-identifier to select the same key—in fact, we do not even want them to know each other's keys because then a customer would be able to decrypt the sensitive attributes of some other customers, which is undesirable. On the other hand, we must ensure that the key share provided by a customer can be used in the recovery of the key of every customer having the same quasi-identifier.

We resolve this dilemma by assuming a $(2N, k)$-*Shamir secret sharing* [28] of a "seed" key x, where each customer i has two shares x_{2i-1} and x_{2i} of the seed key. (Note the meaning of the two parameters of Shamir secret sharing: $2N$ is the overall number of shares and k is the threshold number of shares needed to recover x.) Specifically, there exists a degree-$(k-1)$ polynomial $\mathcal{P}()$ such that $\mathcal{P}(0) = x$. The shares owned by customer i are $x_{2i-1} = \mathcal{P}(2i-1)$ and $x_{2i} = \mathcal{P}(2i)$. A very useful property of Shamir secret sharing is that with k or more shares one can easily derive all other shares using Lagrange interpolation, while with fewer than k shares one has no information about any other shares at all.

The key that we use to encrypt the sensitive attributes $(a_1^{(i)}, \ldots, a_n^{(i)})$ is $H(s_1^{(i)}, \ldots, s_m^{(i)})^{x_{2i-1}}$, where H is a cryptographic hash function. Clearly, this key can be derived if and only if for k or more values of j, $H(s_1^{(i)}, \ldots, s_m^{(i)})^{x_j}$ is available. The key share submitted by customer i is $H(s_1^{(i)}, \ldots, s_m^{(i)})^{x_{2i}}$. Consequently, this submitted key share can actually be used in the recovery of any keys used to encrypt the sensitive attributes where the quasi-identifiers are equal to $(s_1^{(i)}, \ldots, s_m^{(i)})$. These key can be recovered successfully if and only if there are at least k customers with quasi-identifier equal to $(s_1^{(i)}, \ldots, s_m^{(i)})$. Furthermore, even the customers having the same quasi-identifier cannot figure out each other's key because they do not know other customers' shares of the seed key.

4.1 The Protocol

Let S be a security parameter, let p, q be two S-bit primes such that $p = 2q + 1$, let G_q be the quadratic residue subgroup of $\mathbb{Z}_p^\times$ (the multiplicative group mod p), and let H be a cryptographic hash function with range G_q.

Before the protocol starts, we assume that a "seed" key $x \in [0, q-1]$ is shared among customers using $(2N, k)$-Shamir secret sharing and that each customer i has two shares, x_{2i-1} and x_{2i}. Specifically, there exists a degree-$(k-1)$ polynomial $\mathcal{P}()$ such that $x = \mathcal{P}(0)$ and $\forall i \in [1, 2N]$, $x_i = \mathcal{P}(i)$.

DATA SUBMISSION. Customer i encrypts $(a_1^{(i)}, \ldots, a_n^{(i)})$ using $y_{2i-1} = H(s_1^{(i)}, \ldots, s_m^{(i)})^{x_{2i-1}}$ as a symmetric key. Then she sends the miner the ciphertext together with $(s_1^{(i)}, \ldots, s_m^{(i)})$ and $y_{2i} = H(s_1^{(i)}, \ldots, s_m^{(i)})^{x_{2i}}$.

DATA PROCESSING. When the miner has collected all customers' messages, he counts the number of rows (customers) for each different value of $(s_1, \ldots, s_m)$. If for a value of $(s_1, \ldots, s_m)$ there are k or more rows, then he decrypts the sensitive attributes of these rows as follows: let I be a subset of k such rows; the miner computes customer j's symmetric key using

$$y_{2j-1} = \prod_{i \in I} y_{2i}^{\prod_{\ell \neq i, \ell \in I} (2j-1-2\ell)/(2i-2\ell)}.$$

Then the miner decrypts the sensitive attributes of these rows using the computed keys.

4.2 Privacy Analysis

We show our privacy guarantee under a standard cryptographic assumption, the Decisional Diffie-Hellman (DDH) assumption. (See [10] for a survey of DDH.)

THEOREM 4. *Under the DDH assumption and in the random oracle model*[2]*, the protocol for extracting* $\mathcal{K}(T)$ *is ideally private.*

[2]The random oracle model is a methodology frequently used in proofs of security for systems using hash functions. Effectively it makes the assumption that the use of the hash function does not introduce any insecurity.

PROOF. We only need to construct a simulator M for the miner, because the customers do not receive any messages from the miner (and therefore the simulator that simply outputs the party's data, preloaded key shares if any, and coin flips (and no messages) is a valid simulator).

M picks $x' \in [0, q-1]$ uniformly at random and computes $2N$ Shamir shares of x': $x'_1, \ldots, x'_{2N}$.

If customer i's row is in $\mathcal{K}(T)$, then M computes $y'_{2i-1} = H(s_1^{(i)}, \ldots, s_m^{(i)})^{x'_{2i-1}}$, $y'_{2i} = H(s_1^{(i)}, \ldots, s_m^{(i)})^{x'_{2i}}$, and encrypts $(a_1^{(i)}, \ldots, a_n^{(i)})$ using symmetric key y_{2i-1}. M simulates customer i's message with the above symmetric encryptions, $(s_1^{(i)}, \ldots, s_m^{(i)})$, and y'_{2i}.

If customer i's row is not in $\mathcal{K}(T)$, then M still computes $y'_{2i} = H(s_1^{(i)}, \ldots, s_m^{(i)})^{x'_{2i}}$. M simulates customer i's message with a random ciphertext in the symmetric encryption scheme, $(s_1^{(i)}, \ldots, s_m^{(i)})$, and y'_{2i}.

The proof of computational indistinguishability is notationally too complicated to be included in this paper. However, we prove a simplified version of the indistinguishability result as Lemma 5. It is conceptually trivial (though notationally challenging) to extend Lemma 5 to the indistinguishability needed here. □

Below is a simplified version of the indistinguishability result needed in the proof of Theorem 4.

LEMMA 5. *Under the DDH assumption,*

$$\{g_1, g_2, g_3, g_1^{e_1}, g_2^{e_2}, g_3^{\alpha e_1 + \beta e_2}\}_q \stackrel{\mathsf{c}}{\equiv} \{g_1, g_2, g_3, g_1^{e_1}, g_2^{e_2}, g_3^{e_3}\}_q,$$

where g_1, g_2, g_3 are picked uniformly and independently from G_q, and e_1, e_2, e_3 are picked uniformly and independently from $[0, q-1]$.

PROOF. Suppose by way of contradiction that the above indistinguishability result does not hold. Then there exist a probabilistic polynomial-time algorithm $\mathcal{D}$ and a polynomial $f()$ such that, for infinitely many q,

$$|\Pr[\mathcal{D}(g_1, g_2, g_3, g_1^{e_1}, g_2^{e_2}, g_3^{\alpha e_1+\beta e_2}, q) = 1] - \Pr[\mathcal{D}(g_1, g_2, g_3, g_1^{e_1}, g_2^{e_2}, g_3^{e_3}, q) = 1]| \geq 1/f(S).$$

Thus we construct another polynomial-time algorithm $\mathcal{D}'()$ such that

$$\mathcal{D}'(E_1, E_2, E_3, q, g) \stackrel{\mathsf{def}}{=} \mathcal{D}(g, E_2^{e'}, g^{e''}, E_1^{1/\alpha}, E_3^{-e'/\beta}, 1, q),$$

where e', e'' are picked uniformly and independently from $[0, q-1]$. Then clearly, for $\hat{e}_1, \hat{e}_2$ uniformly and independently picked from $[0, q-1]$, we have

$$\begin{aligned}
&\mathcal{D}'(g_1^{\hat{e}_1}, g_1^{\hat{e}_2}, g_1^{\hat{e}_1\hat{e}_2}, q, g_1) \\
&= \mathcal{D}(g_1, g_1^{\hat{e}_2 e'}, g_1^{e''}, g_1^{\hat{e}_1/\alpha}, g_1^{-\hat{e}_1\hat{e}_2 e'/\beta}, 1, q), \\
&= \mathcal{D}(g_1, g_1^{\hat{e}_2 e'}, g_1^{e''}, g_1^{\hat{e}_1/\alpha}, (g_1^{\hat{e}_2 e'})^{-\hat{e}_1/\beta}, (g_1^{e''})^0, q) \\
&= \mathcal{D}(g_1, g_2, g_3, g_1^{\hat{e}_1/\alpha}, g_2^{-\hat{e}_1/\beta}, g_3^0, q).
\end{aligned}$$

The last identity holds because $g_1^{e_2 e'}$ and $g_1^{e''}$ are independent and uniform and we can rename them as g_2 and g_3. We then further rename $\hat{e}_1/\alpha$ and $-\hat{e}_1/\beta$ as e_1 and e_2 and get

$$\mathcal{D}'(g_1^{\hat{e}_1}, g_1^{\hat{e}_2}, g_1^{\hat{e}_1\hat{e}_2}, q, g_1) = \mathcal{D}(g_1, g_2, g_3, g_1^{e_1}, g_2^{e_2}, g_3^{\alpha e_1+\beta e_2}, q).$$

Similarly, we can get, for $\hat{e}_1, \hat{e}_2, \hat{e}_3$ uniformly and independently picked from $[0, q-1]$,

$$\mathcal{D}'(g_1^{\hat{e}_1}, g_1^{\hat{e}_2}, g_1^{\hat{e}_3}, q, g_1) = \mathcal{D}(g_1, g_2, g_3, g_1^{e_1}, g_2^{e_2}, g_3^{e_3}, q).$$

Therefore,

$$|\Pr[\mathcal{D}'(g_1^{\hat{e}_1}, g_1^{\hat{e}_2}, g_1^{\hat{e}_1\hat{e}_2}, q, g_1) = 1] - \Pr[\mathcal{D}'(g_1^{\hat{e}_1}, g_1^{\hat{e}_2}, g_1^{\hat{e}_3}, q, g_1) = 1]| \geq 1/f(S),$$

an obvious contradiction to DDH. □

5. OUR SOLUTION FOR FORMULATION 2

In this section, we solve the second formulation of the problem. Specifically, we provide a protocol that privately k-anonymizes a table by suppressing entries. Our protocol is based on Meyerson and Williams's algorithm (which we refer to as MW) for k-anonymizing a database [24]. Our solution can be viewed as a distributed, privacy-preserving, version of their algorithm. Our protocol provides quantifiable, though not ideal, privacy. Namely, it keeps all information about the suppressed entries private from each individual party, except revealing the distance between each pair of rows.

Our protocol consists of three phases. In the first phase, the protocol allows the miner to compute the distance between each pair of rows. In the second phase, the miner uses the MW algorithm to compute a *k-partition* of the table. (A k-partition is a collection of disjoint subsets of rows in which each subset contains at least k rows and the union of these subsets is the entire table.) In the third phase, the protocol allows the miner to compute the k-anonymized table. The second phase is a direct computation of part of MW (which relies only on the inter-row distances already known to the miner).

We now overview the more complex first and third phases; we describe all three phases in complete detail in Section 5.1.

Design of Phase 1

Recall that the distance between two rows is the number of quasi-identifier attributes in which the rows have different values [24]. If we define

$$\sigma_j^{(i,i')} = \begin{cases} 1 & \text{if } s_j^{(i)} = s_j^{(i')} \\ r & \text{if } s_j^{(i)} \neq s_j^{(i')} \end{cases}$$

(where r is a random element uniformly picked from an exponentially large prime-order cyclic group), then with all but negligible probability the distance between the ith and i'th rows equals $|\{j : \sigma_j^{(i,i')} \neq 1, j \in [1, m]\}|$ (because the the probability of $r = 1$ is negligible). To compute this number, the miner first computes encryptions of $\sigma_j^{(i,i')}$s from encryptions of quasi-identifier attributes; then, a customer rerandomizes and repermutes these encryptions (so that the miner does not learn the value of any specific $\sigma_j^{(i,i')}$ when they are decrypted); finally, the customers jointly help the miner to decrypt the $\sigma_j^{(i,i')}$s.

To allow the miner to compute encryptions of $\sigma_j^{(i,i')}$s, we use the fact that, since the cyclic group mentioned above is of a

prime order,

$$\sigma_j^{(i,i')} = (s_j^{(i)}/s_j^{(i')})^{e_{i,i',j}}, \quad (1)$$

where $e_{i,i',j}$ is a uniformly random exponent. This technique was first used in [7]. (Equation (1) holds because, if $s_j^{(i)} \neq s_j^{(i')}$, then $s_j^{(i)}/s_j^{(i')} \neq 1$. Any element of a prime-order cyclic group not equal to 1 is a generator; and a generator raised to a uniformly random exponent must be a uniformly random element of the cyclic group.) When all quasi-identifier attributes are encrypted using a *multiplicatively homomorphic* encryption scheme (where an encryption of the product of multiple elements can be computed from the encryptions of these elements), it is easy for the miner to compute the encryption of $\sigma_j^{(i,i')}$s using the encryptions of the quasi-identifier attributes. Specifically, in our protocol, we use the ElGamal encryption scheme [15]: an encryption of plaintext $\mathcal{M} \in G_q$ is $\mathcal{C} = (\mathcal{M}y^r, g^r)$, where g is a generator of G_q, $y = g^x$ is the public key (and x is the private key), and r is picked uniformly at random from $[0, q-1]$. To decrypt an ElGamal ciphertext, one simply divides its first component by its second component raised to the secret key.

The remaining question is how the customers jointly help the miner to decrypt ciphertexts of $\sigma_j^{(i,i')}$s. We use a *threshold cryptography* technique similar to that of Desmedt and Frankel [13]. Assume that the private key is shared among the customers using a (N, t)-Shamir secret sharing [28], where t is an arbitrary threshold. (We discuss how to choose t in Section 6.) Then a customer can compute a "partial decryption" by raising the second component of an ElGamal ciphertext to her share of the private key. To compute the plaintext, the miner only needs to take t partial decryptions and interpolate them.

Design of Phase 3

Let $\mathcal{P}$ be the k-partition computed in the second phase. Let $P_\ell \in \mathcal{P}$. Suppose that ith row is in P_ℓ. According to MW, customer i should replace $s_j^{(i)}$ with $*$ if and only if

$$\exists i' \in P_\ell, s_j^{(i)} \neq s_j^{(i')}.$$

With high probability, this is equivalent to

$$\prod_{i' \in P_\ell, i' \neq i} \sigma_j^{(i,i')} \neq 1.$$

Because ElGamal is multiplicatively homomorphic, it is easy to compute an encryption of $\prod_{i' \in P_\ell, i' \neq i} \sigma_j^{(i,i')}$. Hence the remaining technical question is how other customers jointly help customer i to decrypt it. To achieve this goal, we again use the technique of partial decryptions in the first phase; the main difference is that customer i only needs the help of $t-1$ other customers, because customer i herself already has a share of the private key.

5.1 The Protocol

We now give a detailed description of the entire protocol. Suppose that S is a security parameter, that p, q are S-bit primes such that $p = 2q + 1$, and that G_q is the quadratic residue subgroup of $\mathbb{Z}_p^\times$. Let $t \in [2, N-1]$ be a threshold. In this section, we assume that N customers share a private key $x \in [0, q-1]$ using (N, t)-Shamir secret sharing, where customer i's share is denoted by x_i. Specifically, there exists a degree-$(t-1)$ polynomial $\mathcal{P}()$ such that $x = \mathcal{P}(0)$ and $\forall i \in [1, N]$, $x_i = \mathcal{P}(i)$. We also assume that the corresponding public key $y = g^x$ (where g is a generator of G_q) is known to all involved parties (customers and the miner).

5.1.1 Phase 1

In this phase, the miner computes the distance between every pair of rows, following the method overviewed above.

Submission of Encrypted Quasi-identifier Attributes. Each customer i encrypts each of her quasi-identifier attributes using ElGamal with public key y (for $j = 1$ to m):

$$\overline{s}_j^{(i)} = (s_j^{(i)} y^{r_{ij}}, g^{r_{ij}}),$$

where r_{ij} is picked uniformly at random from $[0, q-1]$. Then the customers send all these encryptions to the miner. The ciphertext $\overline{s}_j^{(i)}$ above has two components; we denote the first and the second components by $\overline{s}_j^{(i)}\langle 1\rangle$ and $\overline{s}_j^{(i)}\langle 2\rangle$ respectively.

Computing Encryptions of $\sigma_j^{(i,i')}$. For each pair (i, i'), the miner computes the quotients of their corresponding quasi-identifier attributes: (for $j = 1$ to m)

$$q_j^{(i,i')} = (\overline{s}_j^{(i)}\langle 1\rangle / \overline{s}_j^{(i')}\langle 1\rangle, \overline{s}_j^{(i)}\langle 2\rangle / \overline{s}_j^{(i')}\langle 2\rangle).$$

Then the miner raises the quotients to random powers: (for $j = 1$ to m)

$$p_j^{(i,i')} = ((q_j^{(i,i')}\langle 1\rangle)^{e_{i,i',j}}, (q_j^{(i,i')}\langle 2\rangle)^{e_{i,i',j}}),$$

where $e_{i,i',j}$ is picked uniformly at random from $[0, q-1]$.

Rerandomization and Repermutation. The miner sends $\{p_j^{(i,i')} : i, i' \in [1, N], i \neq i', j \in [1, m]\}$ to an arbitrary customer i_0. Customer i_0 rerandomizes each $p_j^{(i,i')}$ and repermutes $(p_1^{(i,i')}, \ldots, p_m^{(i,i')})$ for each pair (i, i'). Denote the result of the above rerandomization and repermutation operations by $\{u_j^{(i,i')} : i, i' \in [1, N], i \neq i', j \in [1, m]\}$. Then customer i_0 sends $\{u_j^{(i,i')} : i, i' \in [1, N], i \neq i', j \in [1, m]\}$ back to the miner.

Decrypting $\sigma_j^{(i,i')}$. Consider a set I of t customers, where $i_0 \notin I$. To each of these t customers, the miner sends $\{u_j^{(i,i')}\langle 2\rangle : i, i' \in [1, N], i \neq i', j \in [1, m]\}$. Each of the picked customer $i'' \in I$ raises all elements she receives to the $x_{i''}$th power: (for each (i, i', j) such that $i, i' \in [1, N], i \neq i', j \in [1, m]$)

$$v_{j,i''}^{(i,i')} = (u_j^{(i,i')}\langle 2\rangle)^{x_{i''}}.$$

Then she sends $\{v_{j,i''}^{(i,i')} : i, i' \in [1, N], i \neq i', j \in [1, m]\}$ back to the miner.

Finally, the miner computes

$$\hat{\sigma}_j^{(i,i')} = u_j^{(i,i')}\langle 1\rangle / \prod_{i'' \in I} (v_{j,i''}^{(i,i')})^{\Pi_{\ell \neq i'', \ell \in I}\, \ell/(\ell - i'')}.$$

Note that $\hat{\sigma}_1^{(i,i')}, \ldots, \hat{\sigma}_m^{(i,i')}$ are nothing but a permutation of $\sigma_1^{(i,i')}, \ldots, \sigma_m^{(i,i')}$; thus $|\{j : \sigma_j^{(i,i')} \neq 1, j \in [1, m]\}| = |\{j :$

$\hat{\sigma}_j^{(i,i')} \neq 1, j \in [1,m]\}|$. For each pair (i,i'), the miner counts $|\{j : \hat{\sigma}_j^{(i,i')} \neq 1, j \in [1,m]\}|$. The distance between the ith and i'th rows is equal to this number.

5.1.2 Phase 2

In this phase, knowing the pairwise distances of the rows, the miner follows the first part of the MW algorithm to compute a k-partition $\mathcal{P} = \{P_1, \ldots, P_L\}$.

5.1.3 Phase 3

In this phase, the miner computes the k-anonymized table with the help of the customers, as overviewed above.

COMPUTING ENCRYPTIONS OF $\prod_{i' \in \mathcal{P}_l, i' \neq i} \sigma_j^{(i,i')}$. For each $P_\ell \in \mathcal{P}$, each $i \in P_\ell$, each $j \in [1,m]$, the miner computes

$$p_j^{(i)} = (\prod_{i' \in P_\ell, i' \neq i} p_j^{(i,i')}\langle 1\rangle, \prod_{i' \in P_\ell, i' \neq i} p_j^{(i,i')}\langle 2\rangle).$$

DECRYPTING $\prod_{i' \in \mathcal{P}_\ell, i' \neq i} \sigma_j^{(i,i')}$. Then, for each P_ℓ, let I_ℓ be a set of $t-1$ customers such that $i_0 \notin I_\ell$ and $I_\ell \cap P_\ell = \emptyset$. The miner sends $\{p_j^{(i)}\langle 2\rangle : i \in P_\ell, j \in [1,m]\}$ to every customer in I_ℓ. Each customer $i' \in I_\ell$ computes (for each $i \in P_\ell$ and each $j \in [1,m]$)

$$w_j^{(i,i')} = (p_j^{(i)}\langle 2\rangle)^{x_{i'}},$$

and sends $\{w_j^{(i,i')} : i \in P_\ell, j \in [1,m]\}$ back to the miner.

The miner computes (for each P_ℓ, each $i \in P_\ell$ and each $j \in [1,m]$)

$$z_j^{(i)} = p_j^{(i)}\langle 1\rangle / \prod_{i' \in I_\ell} (w_j^{(i,i')})^{\prod_{i'' \in I_\ell, i'' \neq i'} i''/(i''-i')}.$$

He sends $\{z_j^{(i)} : j \in [1,m]\}$, $\{p_j^{(i)}\langle 2\rangle : j \in [1,m]\}$, and P_ℓ to each customer i.

For each $j \in [1,m]$, each customer $i (\in P_\ell)$ computes

$$\hat{z}_j^{(i)} = p_j^{(i)}\langle 2\rangle^{x_i \prod_{i'' \in I_\ell, i'' \neq i} i''/(i''-i)},$$

and compares $\hat{z}_j^{(i)}$ with $z_j^{(i)}$. If they are equal, then customer i sets $\hat{s}_j^{(i)} = s_j^{(i)}$; otherwise, customer i sets $\hat{s}_j^{(i)} = *$. Finally, customer i sends the miner $(\hat{s}_1^{(i)}, \ldots, \hat{s}_m^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)})$.

5.2 Privacy Analysis

Our protocol leaks only the distance between each pair of rows. Let $\mathsf{Distance}(T, i, i')$ denote the distance between the ith and i'th rows of table T.

THEOREM 6. *The protocol of k-anonymization by suppressing entries leaks only $\{\mathsf{Distance}(T, i, i') : i, i' \in [1,N], i \neq i'\}$, under the DDH assumption.*

PROOF. We first construct a simulator M for the miner. For the phase of computing pairwise distances, M simulates the first-round messages the miner receives using mN random ElGamal ciphertexts. For the second-round messages the miner receives (which are from i_0), M simulates each $u_j^{(i,i')}$ using a random ElGamal ciphertext $u'^{(i,i')}_j$. Then, M picks $i_0'' \in I$; for each $i'' \neq i_0''$, $i'' \in I$, M simulates the third round message $v_{j,i''}^{(i,i')}$ using a random element $v'^{(i,i')}_{j,i''}$ of G_q. To simulates $v_{j,i_0''}^{(i,i')}$, M first sets the values of $\sigma'^{(i,i')}_j$: for each pair (i,i'), the m variables $\{\sigma'^{(i,i')}_j : j \in [1,m]\}$ have exactly $m - \mathsf{Distance}(T,i,i')$ 1's; M randomly picks this number of variables and sets them to 1 and then sets all the remaining variables randomly. After that, M simulates $v_{j,i_0''}^{(i,i')}$ using

$$v'^{(i,i')}_{j,i_0''} = \frac{u'^{(i,i')}_j\langle 1\rangle}{\sigma'^{(i,i')}_j \prod_{i'' \in I, i'' \neq i_0''} (v'^{(i,i')}_{j,i''})^{\prod_{\ell \neq i'', \ell \in I} \frac{\ell}{\ell - i''}}}.$$

For the phase of computing anonymized data, M simulates the first-round messages the miner receives using $Nm(t-1)$ independent random elements of G_q. M simulates the last-round messages using the rows in $\mathsf{Anonymized}(T)$.

The computational indistinguishability follows from the semantic security of ElGamal encryption, which is well known to hold under DDH [10].

Now we construct a simulator M_i for customer i. If $i = i_0$, then M_i simulates the messages received by i_0 using $N(N-1)m$ random ElGamal ciphertexts. If $i \in I$, then M_i simulates the messages received by i in the phase of computing pairwise distances using $N(N-1)m$ independent random elements of G_q. If $i \in I_\ell$, then M_i simulates the first-round messages received by i in the phase of computing anonymized data using independent random elements of G_q. For any customer i, for the last-round messages customer i receives, M_i simulates each $p_j^{(i)}\langle 2\rangle$ using an independent random element $p'^{(i)}_j$ of G_q; if the (i,j)-entry of the anonymized sensitive attributes is $*$, then M_i simulates $z_j^{(i)}$ using an independent random element of G_q as well; otherwise, M_i simulates $z_j^{(i)}$ using

$$z'^{(i)}_j = (p'^{(i)}_j)^{x_i \prod_{i'' \in I_\ell, i'' \neq i} i''/(i''-i)}.$$

The computational indistinguishability follows immediately from the semantic security of ElGamal encryption. □

6. DISCUSSION

In this paper, we studied methods for creating k-anonymous tables in a distributed scenario without the need for a central authority and while maintaining customer privacy. We formulate the problem in two ways.

For the first problem formulation, a protocol must extract the k-anonymous part of a table. The major advantage of our protocol is that it is non-interactive—each customer only sends a single flow of communication to the miner. Therefore, customers can "submit data and go". Another advantage is that the solution is very efficient. The dominating computational overhead of each customer is two modular exponentiations; the dominating computational overhead of the miner is kN_k modular exponentiations, where N_k is the number of rows in the k-anonymous part of the table. The limitation of this problem formulation is that it is suitable only if the original table is already close to k-anonymous, as otherwise

the subset of the table learned by the miner may not be of sufficient utility.

For the second problem formulation, a protocol k-anonymizes a table by suppressing entries. The advantage of this approach is that it can produce useful results even when the original table is not close to k-anonymous. Our solution to this problem formulation leaks a small amount of information beyond the k-anonymous result—namely, the distance between each pair of rows. Consequently, this approach is a good choice for the applications in which revealing the distances between rows can be tolerated.

We have shown that our solutions protect privacy against any individual party involved. The solutions can also be extended to provide privacy even if some parties collude and pool their information: up to $k-1$ (for the first protocol) or $t-1$ (for the second protocol). In the second protocol, this requires a slight change so that the task of customer i_0 is distributed among t customers. The choice of threshold t is subject to a trade-off: the greater the value of t, the more colluding parties the protocol can work against, but the more expensive the protocol is. In a practical application, an appropriate value of t must be chosen according to the application's requirements of privacy and efficiency.

7. REFERENCES

[1] J. O. Achugbue and F. Y. Chin. The effectiveness of output modification by rounding for protection of statistical databases. *INFOR*, 17(3):209–218, 1979.

[2] N. Adam and J. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Survey*, 21(4):515–556, 1989.

[3] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *Proceedings of 9th International Conference on Extending Database technology*. Springer, 2004.

[4] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. k-anonymity: Algorithms and hardness. Under review, 2004.

[5] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255, 2001.

[6] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of 19th ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[7] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology - Proceedings of EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer-Verlag, 2001.

[8] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of 21st International Conference on Data Engineering*, 2005.

[9] L. L. Beck. A security mechanism for statistical databases. *ACM Transactions on Database Systems*, 5(3):316–338, September 1980.

[10] D. Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory, Third International Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998.

[11] F. Y. Chin and G. Ozsoyoglu. Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, SE-8(6):113–139, April 1982.

[12] T. Dalenius. Finding a needle in a haystack—or identifying anonymous census record. *Journal of Official Statistics*, 2(3):329–336, 1986.

[13] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology - Proceedings of CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, 1990.

[14] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210. ACM Press, 2003.

[15] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology - Proceedings of CRYPTO 84*, pages 10–18, 1985.

[16] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 211–222. ACM Press, 2003.

[17] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228. ACM Press, 2002.

[18] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure applications of Pedersen's distributed key generation protocol. In *CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 373–390, 2003.

[19] O. Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[20] M. Kantarcioglu and C. Clifton. Privacy preserving distributed mining of association rules on horizontally partitioned data. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 639–644. ACM, 2002.

[21] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of 3rd IEEE International Conference on Data Mining*, Florida, Nov 2003.

[22] J. M. Kleinberg, C. H. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *Proceedings of 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 86–91, 2000.

[23] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[24] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Paris, France, June 2004.

[25] S. Reiss. Practical data swapping: The first steps. *ACM Transactions on Database Systems*, 9(1):20–37, 1984.

[26] P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[27] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, page 188. ACM Press, 1998.

[28] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[29] A. Shoshani. Statistical databases: Characteristics, problems and some solutions. In *Proceedings of 8th International Conference on Very Large Data Bases*, pages 208–222, 1982.

[30] L. Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. In *Proceedings, Journal of the American Medical Informatics Association*, 1997.

[31] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002.

[32] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[33] J. Traub, Y. Yemini, and H. Wozniakowksi. The statistical security of a statistical database. *ACM Transactions on Database Systems*, 9(4):672–679, 1984.

[34] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, 2002.

[35] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215. ACM Press, 2003.