Anonymity-Preserving Data Collection*

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

ABSTRACT

Protection of privacy has become an important problem in data mining. In particular, individuals have become increasingly unwilling to share their data, frequently resulting in individuals either refusing to share their data or providing incorrect data. In turn, such problems in data collection can affect the success of data mining, which relies on sufficient amounts of accurate data in order to produce meaningful results. Random perturbation and randomized response techniques can provide some level of privacy in data collection, but they have an associated cost in accuracy. Cryptographic privacy-preserving data mining methods provide good privacy and accuracy properties. However, in order to be efficient, those solutions must be tailored to specific mining tasks, thereby losing generality.

In this paper, we propose efficient cryptographic techniques for online data collection in which data from a large number of respondents is collected anonymously, without the help of a trusted third party. That is, our solution allows the miner to collect the original data from each respondent, but in such a way that the miner cannot link a respondent's data to the respondent. An advantage of such a solution is that, because it does not change the actual data, its success does not depend on the underlying data mining problem. We provide proofs of the correctness and privacy of our solution, as well as experimental data that demonstrates its efficiency. We also extend our solution to tolerate certain kinds of malicious behavior of the participants.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption

General Terms

Algorithms; Security.

*This work was supported by the National Science Foundation under Grant No. CCR-0331584.

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.

Keywords

Anonymity, Data Collection, Data Mining.

1. INTRODUCTION

With the rapid development of computer and networking technologies, huge amounts of data are collected and analyzed all over the world every day. Some of this data is privacy-sensitive, and issues of privacy protection of sensitive data are receiving more and more attention from the public [19, 25, 26]. In particular, since data mining is a powerful tool for discovering knowledge from large amounts of data, protection of privacy in data mining has become one of the top priorities of study [5].

Currently, a significant amount of data used in data mining is collected on networks. Consider a typical scenario of online data collection: the miner (or data collector) queries large sets of respondents (or customers), and each respondent submits her data to the miner in response. Clearly, this can be an efficient and convenient procedure, assuming the respondents are willing to submit their data. However, the respondents' willingness to submit data is affected by their privacy concerns [6]. For example, the miner may be a medical researcher who studies the relationship between dining habits and a certain disease. Since a respondent may not want to reveal what food she eats and/or whether she has that disease, she may give false information or decline to provide information. Therefore, protecting privacy of respondents is highly important to the success of data mining.

In the example we mentioned above, one possible solution is that the miner collects data *anonymously*. That is, he collects records from the respondents containing each respondent's dining habits and health information related to that disease, but does not know which record came from which respondent. Since a respondent is "hidden" among many peers, she should feel comfortable to submit her data.

We generalize this idea to propose an approach called *anonymity-preserving data collection*. Specifically, we propose that the miner should collect data in such a way that he is unable to link any piece of data collected to the respondent who provided that piece of data. In this way, respondents do not need to worry about their privacy. Furthermore, the data collected is not modified in any way, and thus the miner will have the freedom to apply any suitable mining algorithms to the data.

1.1 Related Work

A variety of methods have been proposed to protect the privacy of each respondent by perturbing the respondents' data. Warner proposed randomized response techniques, together with statistical techniques for reconstructing distributions from the perturbed responses [37]. Different random perturbation methods have been proposed and applied in different data mining algorithms [2, 1, 12, 28, 11, 9]. Random perturbation is very efficient, but in general it can induce a tradeoff between privacy of respondents and accuracy of the data mining result: the more privacy each respondent has, the less accurate the result of mining is, and vice versa. Although some perturbation techniques lead to good accuracy plus good privacy in specific data mining problems, these perturbation techniques would produce inaccurate results when used in other data mining problems. The privacy-preserving properties of the perturbation techniques are further explored in [21].

Privacy-preserving data mining solutions have also been proposed based on cryptographic techniques (c.f. [23, 34, 38, 39]). However, the design of the cryptographic protocol depends on the specific mining task, unless a prohibitively expensive general-purpose secure multiparty computation is used [40, 15].

A class of closely related work studies how to measure privacy in data mining. This includes privacy definitions based on confidence intervals [2], based on mutual information [1], and based on priori and posterior knowledge [11, 8]. Privacy definitions motivated by cryptographic notions of confidentiality are given by Gilburd et al. [13] and Dwork and Nissim [10], respectively.

We note that our problem could be solved if an anonymous communication channel were available. However, building an anonymous channel is a nontrivial task. General-purpose ways to build anonymous channels include mix networks [3, 24, 29, 20, 18], dining cryptographer networks (DC-nets) [4, 36, 17] and k-anonymous message transmission [35], all of which are still under active study. Compared with mix networks, our work can be viewed as a (nontrivial) application of certain techniques in their design to the scenario of distributed data collection, which eliminates the burden of introducing a set of mix servers while keeping analogous security guarantees. Compared with DC-nets and k-anonymous message transmission, our solution produces more efficient results for our setting. For example, the communication cost of k-anonymous message transmission is cubic in the number of users; in comparison, our solutions in this paper have linear communication costs.

Systems like Crowds [27] and Hordes [22] provide practical anonymity for web access without assuming an existing anonymous channel. Similarly, our work provides anonymity for data collection without assuming an existing anonymous channel. However, Crowds and Hordes assume eavesdroppers involved are local (i.e., can only eavesdrop communications within one hop). They do not guarantee anonymity when an adversary is able to eavesdrop communications in the entire system. In contrast, our solutions in this paper can work against global eavesdroppers.

In contrast to our setting, work on k-anonymization [32, 31, 30, 7] deals with the complementary problem of ensuring that data does not reveal the identity of the data subject, by processing a database table to de-associate privacy-sensitive attributes from the corresponding identifiers. We note that the goal of k-anonymization is somewhat different: in k-anonymization, making the data anonymous is the target, but how it is communicated to the data miner in the first place is not addressed; in our work, we assume the data is inherently anonymous (as in our example of dining habits and a disease), and address the problem of making the data submission procedure anonymous. (See Section 2 for the precise definition of the type of anonymity we consider).

1.2 Our Contributions

In this paper, we propose an approach called *anonymity-preserving data collection*. Instead of making each respondent's data oblivious to the miner, our approach reveals all respondents' data to the miner, but does not allow the miner to link any respondent's data to the corresponding respondent. As long as the data itself does not contain information that can be used for identification, this is usually sufficient to protect respondents' privacy. A strong advantage of this approach is its generality: since the data collected is not encrypted or perturbed, the miner can then use the data freely for a variety of data mining algorithms. Our contributions can be summarized as follows:

- We introduce the notion of anonymity-preserving data collection.
- We present a concrete protocol for anonymity-preserving data collection under the assumption that all participants follow the protocol. Proofs are given for its correctness and anonymity property.
- We extend our solutions to provide anonymity protection even if the miner or the miner plus some respondents are acting maliciously.
- We provide experimental results to measure the practical efficiency of our solutions.

In Section 2, we provide a formal definition of anonymity. In Section 3, we give a basic solution in the model that the miner and all respondents follow the protocol. We also prove the correctness and anonymity properties of the protocol and present experimental results of the efficiency. In Section 4, we describe an extended protocol to protect respondents privacy against a malicious data miner, again with experimental results of efficiency. We further extend our protocol to works against a malicious miner plus some malicious respondents in Section 5.

2. PRELIMINARIES

In this section, we give a specification of our data collection problem as well as a formal definition of our anonymity requirements.

2.1 The Data Collection Problem

We consider a scenario of data collection in which there is a data miner and a (potentially large) number of respondents. Each respondent owns a piece of data and the miner intends to mine the data so that he can find useful patterns. In our setting, the goal is to allow the miner to collect the data without being able to determine which piece of data came from whom. Specifically, let N be a *small* constant number, typically 20, 50, or 100. We divide the respondents into groups, where each group has N members. In order to make the protocol practical, the miner collects the data from one group at a time, so that only N respondents need to be simultaneously online.[1] Our requirement is that each respondent should be "hidden" in the N respondents in her group. In other words, the miner should get N pieces of data from a group, but should not know which piece came from which group member.

In the sequel, we restrict our discussion to one group of respondents and denote the N respondents in this group by $1, \ldots, N$. We assume that there is a private and authenticated communication channel between each respondent and the miner. (Note that such communication channels can be implemented using standard protocols like SSL and IPSec. We do not go into details of the implementation of channels in this paper.) Although communication channels between respondents may exist in some practical situations, to make the problem as general as possible, we do not assume their existence in our problem. Figure 1 illustrates the resulting layered architecture of our solution. Our anonymity-preserving data collection solution runs over any private and authenticated channels between respondents and miner, and the miner can use any data mining tools thereafter.

We denote by d_i the piece of data owned by respondent i, whose length is bounded by a security parameter κ. There are two possible ways for the miner to violate respondent i's anonymity: either d_i contains some information about respondent i (like respondent i's identifier, telephone number, or zip code) and by looking at this information the miner is able to associate d_i with respondent i, or during the data collection process the miner observes that d_i comes from respondent i. As noted before, in this paper we focus on the second possibility. In particular, we assume that d_i does not contain information that can be used to associate it with respondent i;[2] in this case, respondent i will remain anonymous as long as the data collection process preserves her anonymity.

Data Mining Tools
Anonymity-Preserving Data Collection **(This paper)**
Private and Authenticated Communication Channel
Database and Operating System

Figure 1: System Components

2.2 Formal Definition of Anonymity

In this paper, we allow for the possibility that some of the respondents may be corrupted and colluding with the miner. Informally, the data collection process preserves each honest respondent's anonymity if, when we arbitrarily switch the data between honest respondents, the miner (with the help of dishonest respondents) cannot see any difference in the data collection process. Mathematically, let σ be an arbitrary permutation on $\{1, \ldots, N\}$; then $(d_{\sigma(1)}, \ldots, d_{\sigma(N)})$ is the result of arbitrarily switching data between respondents. We further require that $\sigma(i) = i$ for any corrupted respondent i, which means only the honest respondents' data are switched. The anonymity requirement is that the miner cannot distinguish the data collection procedure in which the respondents have data $(d_1, \ldots, d_N)$ from the data collection procedure in which respondents have data $(d_{\sigma(1)}, \ldots, d_{\sigma(N)})$.

More formally, we give the following definition of anonymity in a standard cryptographic model, the *semi-honest model* [14]. In the semi-honest model, the miner and the corrupted respondents follow the protocol but attempt to derive private information and violate the anonymity of the honest respondents.

DEFINITION 1. *A protocol for the data collection problem preserves each honest respondent's anonymity against the miner and $t-1$ corrupted respondents in the semi-honest model if, for any $I \subseteq \{1, \ldots, n\}$ such that $|I| = t-1$, for any $(d_1, \ldots, d_N)$ and any permutation σ on $\{1, \ldots, N\}$ such that $\forall i \in I, \sigma(i) = i$,*

$$\begin{aligned} & \{\mathsf{view}_{\mathsf{miner},I}(d_1, \ldots, d_N)\} \\ \stackrel{c}{\equiv}\ & \{\mathsf{view}_{\mathsf{miner},I}(d_{\sigma(1)}, \ldots, d_{\sigma(N)})\}. \end{aligned}$$

In the above, $\stackrel{c}{\equiv}$ denotes *computational indistinguishability* and $\{\mathsf{view}_{\mathsf{miner},I}(d_1, \ldots, d_N)\}$ is the joint *view* of the miner and the set I of $t-1$ corrupted respondents. (See a standard book of cryptography, e.g., [14] for the formal definitions of these concepts.) Intuitively, Definition 1 states that the adversary (i.e., the miner and the corrupted respondents) cannot notice any difference in his view if we arbitrarily switch data between the honest respondents. Therefore, the miner and the corrupted respondents jointly learn nothing about which piece of data corresponds to which honest respondent. Clearly, this is consistent with the intuitive understanding of anonymity.

In this paper, we not only develop a solution in the semi-honest model, but also extend the solution to another standard cryptographic model, the *malicious model* [14]. Because defining anonymity in the malicious model is significantly more complicated and requires many details out of the scope of this paper, we do not formalize a definition of anonymity in the malicious model. Instead, we give an informal explanation of anonymity in this model as used in our setting: first, an anonymity-preserving protocol in the malicious model needs to be anonymity-preserving when all participants follow the protocol; second, when any malicious participant deviates from the protocol, the honest participants must be able to detect this before the anonymity is violated, so that the honest participants can abort the protocol without their anonymity being compromised. In this way, the malicious participants are effectively "forced" to follow the protocol.

3. THE BASIC SOLUTION

In this section, we give a solution to the problem of anonymity-preserving data collection in the semi-honest model. Extensions of this solution to the malicious model are presented in Sections 4 and 5.

3.1 Overview of the solution

Our goal is that the miner should obtain a random permutation of the respondents' data $(d_1, \ldots, d_N)$, without knowing which piece of data comes from which respondent. To achieve this goal, we use *ElGamal encryption* (further described below) together with a *rerandomization* technique and a *joint decryption* technique. Both of these techniques

[1] Even this requirement can be removed if *certain members of the group* (called "leaders" in our solutions) remain online until the data collection procedure in this group is completed.

[2] This can be achieved by having respondent i remove from her submitted data any information related to her identity.

have been used extensively in mix networks, e.g., [24, 29, 20, 18].

Let G ($|G| = q$, where q is a large prime) be a cyclic group in which the discrete logarithm is hard[3], and let g be a generator of G. The ElGamal encryption scheme uses a key pair (x, y) such that $y = g^x \bmod q$, where x is a private key and y is a public key[4]. In this scheme, to encrypt a message M using the public key y, one computes $C = (My^r, g^r)$, where the exponentiations are done modulo q and r is chosen uniformly at random from $[0, q-1]$. (Throughout the paper, all exponentiations are modulo q.) To decrypt the ciphertext C using the private key x, one computes $M = C^{(1)}/(C^{(2)})^x$, where $C^{(1)}$ and $C^{(2)}$ denote the first and the second components of C, respectively. It has been shown in [33] that (under standard complexity-theoretic assumptions,) the ElGamal encryption scheme is secure in the sense of *semantic security*. (See [16] for the definition of semantic security).

In the ElGamal encryption scheme, one cleartext has many possible encryptions, since the random number r can take on many different values. ElGamal supports rerandomization, which means computing a different encryption of M from a given encryption of M. A related operation is permutation of the order of items, which means randomly rearranging the order of items. If we rerandomize and permute a sequence of ciphertexts, then we get another sequence of ciphertexts with the same multiset of cleartexts but in a different order. Looking at these two sequences of ciphertexts, the adversary cannot determine any information about which new ciphertext corresponds to which old ciphertext.

In our solution, t of the N respondents act as "leaders". Leaders have the special duty of anonymizing the data. At the beginning of the protocol, all respondents encrypt their data using a public key which is the product of all leaders' public keys. Note that the private key corresponding to this public key is the sum of all leaders' private keys; without the help of all leaders, nobody can decrypt any of these encryptions. The leaders then rerandomize these encryptions and permute them. Finally, the leaders jointly help the miner to decrypt the new encryptions, which are in an order independent of the original encryptions.

For notational convenience, we assume in the sequel that the leaders are respondents 1 through t. In practice, the choice of leaders can be arbitrary or can be dependent on the application.

3.2 Respondent Keys

Each respondent i has a key pair (x_i, y_i) ($x_i \in [0, q-1]$, $y_i \in G$) such that $y_i = g^{x_i}$ in G. (All computations throughout this paper take place in G). Here, the public key y_i is known to all participants, while the private key x_i is kept secret by respondent i. In the sequel, let

$$y = \prod_{i=1}^{t} y_i,$$

and

$$x = \sum_{i=1}^{t} x_i.$$

In our protocol, we use this public value y as a public key to encrypt respondent data. Clearly, $y = g^x$. So, decrypting these encryptions of respondent data needs this secret value x, which is not known to any individual respondent.

3.3 Protocol

- Phase 1: Data submission.
  - For $i = 1, \ldots, N$, each respondent i encrypts her data using public key y:

    $$C_i \stackrel{\text{def}}{=} (C_i^{(1)}, C_i^{(2)}) = (y^{r_i} d_i, g^{r_i}),$$

    where r_i is picked uniformly at random from $[0, q-1]$. Then respondent i sends C_i to the miner.

- Phase 2: t-round anonymization. For $i = 1, \ldots, t$, the miner and the respondents work as follows.
  - At the beginning of the ith round, the miner sends $(C_1, \ldots, C_N)$ to leader i.
  - Leader i rerandomizes each piece of data and permutes the pieces: for $j = 1, \ldots, N$,

    $$\begin{aligned} R_j &\stackrel{\text{def}}{=} (R_j^{(1)}, R_j^{(2)}) \\ &= (C_{\pi_i(j)}^{(1)} \cdot y^{\delta_{\pi_i(j)}}, C_{\pi_i(j)}^{(2)} \cdot g^{\delta_{\pi_i(j)}}), \end{aligned}$$

    where π_i is a random permutation on $\{1, \ldots, N\}$, and each δ_j is picked independently and uniformly from $[0, q-1]$.
  - For $j = 1, \ldots, N$, leader i sets $C_j = R_j$. Then leader i sends $(C_1, \ldots, C_N)$ back to the miner.

- Phase 3: Decryption.
  - The miner sends $(C_1^{(2)}, \ldots, C_N^{(2)})$ to all leaders.
  - Each leader i computes partial decryptions: for $j = 1, \ldots, N$,

    $$p_{j,i} = (C_j^{(2)})^{x_i}.$$

  - Each leader i sends the miner the partial decryptions $(p_{1,i}, \ldots, p_{N,i})$.
  - The miner computes the decryptions: for $j = 1, \ldots, N$,

    $$d'_j = C_j^{(1)} / \prod_{i=1}^{t} p_{j,i}.$$

3.4 Correctness

THEOREM 2. *If all participants follow the protocol, then the miner's result $(d'_1, \ldots, d'_N)$ is a permutation of $(d_1, \ldots, d_N)$.*

PROOF. At the end of Phase 1, the miner has received encryptions of $(d_1, \ldots, d_N)$. In Phase 2, these ciphertexts are rerandomized and permuted; therefore, at the end of Phase

[3]The discrete logarithm problem is a standard computational problem used in cryptography. Many cryptographic tools are based on the assumed hardness of the discrete logarithm.

[4]Throughout this paper, by "key" we mean a cryptographic key rather than a database key.

2, $(C_1, \ldots, C_N)$ represents the encryptions of a permutation of $(d_1, \ldots, d_N)$. Since

$$\begin{aligned} d'_j &= C_j^{(1)} / \prod_{i=1}^{t} p_{j,i} \\ &= C_j^{(1)} / \prod_{i=1}^{t} (C_j^{(2)})^{x_i} \\ &= C_j^{(1)} / (C_j^{(2)})^{\sum_{i=1}^{t} x_i} \\ &= C_j^{(1)} / (C_j^{(2)})^{x}, \end{aligned}$$

the cleartexts of $(C_1, \ldots, C_N)$ are $(d'_1, \ldots, d'_N)$. Therefore, $(d'_1, \ldots, d'_N)$ is a permutation of $(d_1, \ldots, d_N)$. $\square$

3.5 Anonymity

THEOREM 3. *The protocol of Section 3.3 preserves the anonymity of each honest respondent against the miner and $t-1$ corrupted respondents in the semi-honest model.*

PROOF. By contradiction. Assume that this protocol does not preserve the anonymity. Based on this protocol, we give a probabilistic polynomial-time algorithm that distinguishes the ElGamal encryptions of two different cleartexts, which contradicts the well known result that ElGamal is semantically secure.

Clearly it suffices to consider the case in which all the $t-1$ corrupted customers are leaders. The above assumption of not preserving anonymity means that there exist $(d_1, \ldots, d_N)$, a permutation σ on $\{1, \ldots, N\}$ such that $\forall i \in I, \sigma(i) = i$, a probabilistic polynomial algorithm D, and a polynomial $f()$ such that for infinitely many κ,

$$\begin{aligned} &\Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_1, \ldots, d_N)) = 1] \\ &- \Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\sigma(1)}, \ldots, d_{\sigma(N)})) = 1] \\ &> 1/f(\kappa). \end{aligned} \tag{3.1}$$

Now we use a *hybrid argument* (see [14]): since σ is a permutation on $\{1, \ldots, N\}$ such that $\forall i \in I, \sigma(i) = i$, we can decompose it to a number of simple permutations where each simple permutation only switches the order of two elements outside I (that are *not* equal). Formally, there exist permutations $\sigma_1, \ldots, \sigma_m$ ($m < N - (t-1)$) on $\{1, \ldots, N\}$ such that for $j = 1, \ldots, m$, $\forall i \in I, \sigma_j(i) = i$ and that

$$\sigma = \sigma_1 \ldots \sigma_m.$$

Define

$$\mathsf{view}_0 = \mathsf{view}_{\mathsf{miner},I}(d_1, \ldots, d_N),$$

and for $j = 1, \ldots, m$,

$$\mathsf{view}_j = \mathsf{view}_{\mathsf{miner},I}(d_{\sigma_1 \ldots \sigma_j(1)}, \ldots, d_{\sigma_1 \ldots \sigma_j(N)}).$$

Then clearly,

$$\mathsf{view}_m = \mathsf{view}_{\mathsf{miner},I}(d_{\sigma(1)}, \ldots, d_{\sigma(N)}).$$

By Equation 3.1, we know there exists $j \in [0, m]$ such that

$$\Pr[D(\mathsf{view}_j) = 1] - \Pr[D(\mathsf{view}_{j+1}) = 1] > \frac{1}{mf(\kappa)}.$$

The above equation is equivalent to

$$\begin{aligned} &\Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\sigma_1 \ldots \sigma_j(1)}, \ldots, d_{\sigma_1 \ldots \sigma_j(N)})) = 1] \\ &- \Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\sigma_1 \ldots \sigma_j \sigma_{j+1}(1)}, \ldots, \\ &d_{\sigma_1 \ldots \sigma_j \sigma_{j+1}(N)})) = 1] \\ &> \frac{1}{mf(\kappa)}. \end{aligned} \tag{3.2}$$

Note a subtle convention of compositions:

$$\sigma_1 \ldots \sigma_j \sigma_{j+1}(i) = \sigma_{j+1}(\sigma_1(\ldots \sigma_j(i))).$$

(If we do not use this convention, we can get the same result by a simple modification of the indices.) Therefore, we can rewrite Equation 3.2 as

$$\begin{aligned} &\Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\ell_1}, \ldots, d_{\ell_N})) = 1] \\ &- \Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\sigma_{j+1}(\ell_1)}, \ldots, d_{\sigma_{j+1}(\ell_N)})) = 1] \\ &> \frac{1}{mf(\kappa)}, \end{aligned} \tag{3.3}$$

where $\ell_i = \sigma_1 \ldots \sigma_j(i)$. Recall that σ_{j+1} only switches the order of two elements outside I that are not equal; suppose that it switches the order of ℓ_α and ℓ_β ($d_{\ell_\alpha} \neq d_{\ell_\beta}$, $\alpha < \beta$, $\alpha, \beta \notin I$). Formally, we have

$$\begin{aligned} d_{\sigma_{j+1}(\ell_\alpha)} &= d_{\ell_\beta}, \\ d_{\sigma_{j+1}(\ell_\beta)} &= d_{\ell_\alpha}, \end{aligned}$$

and that for any $i \neq \alpha$, $i \neq \beta$,

$$d_{\sigma_{j+1}(\ell_i)} = d_{\ell_i}.$$

Below we give a probabilistic polynomial-time algorithm A that distinguishes an ElGamal encryption of d_{ℓ_α} from an ElGamal encryption of d_{ℓ_β}.

On input ciphertext e, A first computes, using the homomorphic property of ElGamal, another ciphertext e' such that the product of the cleartexts of e and e' is equal to $d_{\ell_\alpha} \cdot d_{\ell_\beta}$: A computes a random encryption of $d_{\ell_\alpha} \cdot d_{\ell_\beta}$ and then divides it by e. Then A rerandomizes e' to get e''. Next, A simulates two executions of our protocol; A extracts the view of the adversary generated in each simulated execution and applies D to it. Specifically, during each simulated execution, A simulates the miner using a process that works exactly as described in the protocol. The simulation of other parties is detailed as follows.

In Phase 1 of the protocol, for any i except α and β, A simulates customer i using a process with input d_{ℓ_i}; the process works exactly as described in the protocol. A simulates customers α and β with two processes with ciphertext inputs; these two processes do not encrypt their inputs as described in the protocol but directly send out their inputs to the simulated miner. (Note that this has no impact on the view of the adversary because $\alpha, \beta \notin I$; thus, it has no impact on the output of D which we need.) During the first simulated execution, the simulated customer α starts with ciphertext e and the simulated customer β starts with e''; during the second execution, the simulated customer α starts with ciphertext e'' and the simulated customer β starts with e. Now recall that $t-1$ leaders are dishonest; suppose that the only honest leader is θ.

In Phase 2, the first $\theta - 1$ rounds of anonymization are simulated exactly as described in the protocol. For $i = \theta, \ldots, t$, A chooses a random permutation ρ_i on $[1, \ldots, N]$

and simulates the ciphertexts $(C_1, \ldots, C_N)$ at the end of round i using random encryptions of $(d_{\rho_i(\ell_1)}, \ldots, d_{\rho_i(\ell_N)})$. The corresponding simulated messages and coin flips can be easily computed from these simulated ciphertexts.

In Phase 3, the simulated messages and coin flips can be easily computed from the simulated ciphertexts $(C_1, \ldots, C_N)$ at the end of round t in Phase 2 together with their decryptions $(d_{\rho_t(\ell_1)}, \ldots, d_{\rho_t(\ell_N)})$.

Applying D to the views of the adversary generated in the simulated executions, A can compute

$$\begin{aligned} o_1 =& D(\mathsf{view}_{\mathsf{miner},I}(d_{\ell_1}, \ldots, d_{\ell_{\alpha-1}}, D(e), d_{\ell_{\alpha+1}}, \ldots, \\ & d_{\ell_{\beta-1}}, D(e''), d_{\ell_{\beta+1}}, \ldots, d_{\ell_N})), \end{aligned}$$

and

$$\begin{aligned} o_2 =& D(\mathsf{view}_{\mathsf{miner},I}(d_{\ell_1}, \ldots, d_{\ell_{\alpha-1}}, D(e''), d_{\ell_{\alpha+1}}, \ldots, \\ & d_{\ell_{\beta-1}}, D(e), d_{\ell_{\beta+1}}, \ldots, d_{\ell_N})), \end{aligned}$$

where $D(e)$ denotes the decryption of e. If $o_1 = 1$ and $o_2 = 0$, A outputs 1; if $o_1 = 0$ and $o_2 = 1$, A outputs 0; otherwise A outputs a uniformly random bit.

Now we analyze the probabilities of outputing 1 with input ciphertext of d_{ℓ_α} or d_{ℓ_β}. For convenience, let

$$p_1 = Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\ell_1}, \ldots, d_{\ell_N})) = 1],$$

and

$$p_2 = \Pr[D(\mathsf{view}_{\mathsf{miner},I}(d_{\sigma_{j+1}(\ell_1)}, \ldots, d_{\sigma_{j+1}(\ell_N)})) = 1].$$

When the input ciphertext is an encryption of d_{ℓ_α}, the probability that we have output equals 1 is

$$\Pr[A(d_{\ell_\alpha}) = 1] = p_1(1 - p_2) + p_1 p_2/2 + (1 - p_1)(1 - p_2)/2.$$

When the input ciphertext is an encryption of d_{ℓ_β}, the probability that we have output equals 1 is

$$\Pr[A(d_{\ell_\beta}) = 1] = p_2(1 - p_1) + p_2 p_1/2 + (1 - p_2)(1 - p_1)/2.$$

Combining the above two equations, we have

$$\begin{aligned} & \Pr[A(d_{\ell_\alpha}) = 1] - \Pr[A(d_{\ell_\beta}) = 1] \\ = \; & p_1(1 - p_2) + p_1 p_2/2 + (1 - p_1)(1 - p_2)/2 \\ & -(p_2(1 - p_1) + p_2 p_1/2 + (1 - p_2)(1 - p_1)/2) \\ = \; & p_1 - p_2 \\ > \; & \frac{1}{mf(\kappa)}. \end{aligned}$$

The last inequality is due to Equation 3.3. However, this contradicts the semantic security of ElGamal. □

3.6 Efficiency Analysis and Measurements

In our protocol, the computational overhead of a non-leader respondent is 2 modular exponentiations. The major computational overhead of a leader is $3N + 2$ modular exponentiations. The major computational overhead of the miner is Nt modular multiplications and N modular divisions. The overall communication is at most $(6t + 2)\kappa N$ bits.

To measure the efficiency of our protocol in practice, we implemented it using the OpenSSL libraries[5] and measured the computational overhead. Since the time spent on communication highly depends on the network bandwidth, we

[5] Available at `http://www.openssl.org`.

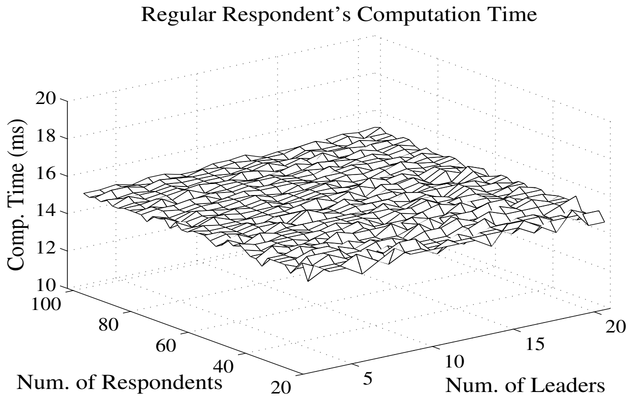

Figure 2: Regular respondent's computation time with semi-honest participants

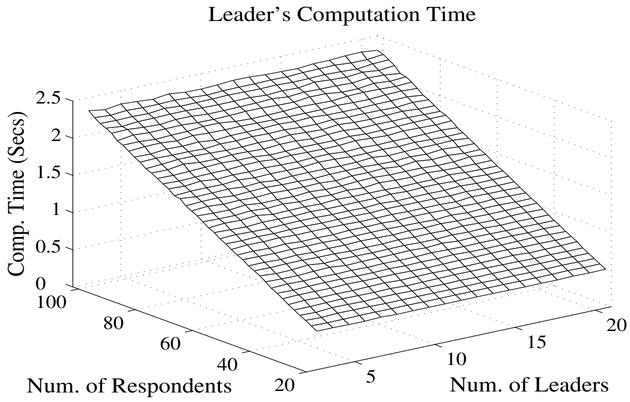

Figure 3: Leader's computation time with semi-honest participants

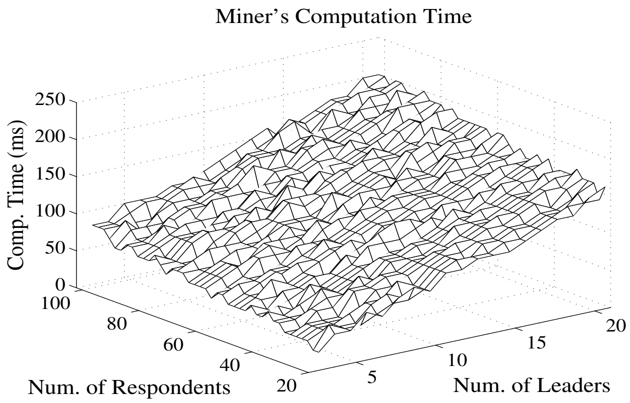

Figure 4: Miner's computation time with semi-honest participants

did not measure the communication overhead in our experiments. In our experiments, the length of cryptographic keys is 1024 bits. The environment used is the NetBSD operating system running on an AMD Athlon 2GHz processor with 512M memory.

We measure the computation times of the three types of participants: regular (i.e., non-leader) respondents, leaders, and the miner. For each of these times, we measure how it varies with different N and t. All our experimental results are consistent with our theoretical analysis.

Figure 2 illustrates our measurements of a regular respondent's computation time: it is always about 15ms regardless of N and t. Figure 3 illustrates our measurements of a leader's computation time: it is linear in N and does not depend on t. For a typical scenario where $N = 20$, the computation time of a leader is about 0.47 seconds. Figure 4 illustrates our measurements of the miner's computation time: it is linear in both N and t. For a typical scenario where $N = 20$ and $t = 3$, the computation time of the miner is about 40ms.

4. MALICIOUS MINER

In this section, we extend our solution to a model in which the miner is malicious and the corrupted respondents are still semi-honest. In Section 5, we study the case in which the corrupted respondents may behave maliciously.

Recall that a *malicious* participant can deviate from the protocol arbitrarily. It is more difficult to preserve anonymity when the miner is malicious. For example, the miner may choose two respondents i and j and replace the encryption of d_j with the encryption of d_i. When the protocol finishes, there will be a piece of data with two copies. The miner can then easily link this piece of data to respondent i. To disallow such behavior and force the miner to follow the protocol, we use a well-known cryptographic tool, *digital signatures*, as we now describe.

4.1 Digital Signatures

A digital signature scheme allows each participant to generate a signature on her message using her private key. Anybody can verify this signature using her public key, but it is infeasible for any other party to forge her signature. Formally, we denote by $s = S_x(M)$ a signature on message M using private key x. We denote by $V_y(M, s)$ the verification function of digital signature using public key y. Thus, for any key pair (x, y) and any message M we have

$$V_y(M, S_x(M)) = \mathsf{accept}.$$

Furthermore, without knowing x it is infeasible to forge a digital signature s such that $V_y(M, s) = \mathsf{accept}$.

Note that in our solution in the semi-honest model, each message sent from the miner to any respondent originally came from a respondent—the miner only forwards the message. Therefore, if the original sender of the message signs it and the receiver of the message verifies the signature, then a cheating miner who deviates from the protocol can be detected.

4.2 Protocol with malicious miner

This protocol assumes that each respondent i has another key pair (x'_i, y'_i) such that $y'_i = g^{x'_i}$ (where x'_i is a private key and y'_i is a public key), in addition to the key pair (x_i, y_i) described in Section 3.2. This new key pair is used for digital signatures.

- Phase 1: Data submission.
  - For $i = 1, \ldots, N$, each respondent i encrypts her data using public key y:
  $$C_i \stackrel{\mathsf{def}}{=} (C_i^{(1)}, C_i^{(2)}) = (y^{r_i} d_i, g^{r_i}),$$
  where r_i is picked uniformly at random from $[0, q-1]$. Respondent i signs C_i using private key x'_i:
  $$s_i = S_{x'_i}(C_i).$$
  Then respondent i sends (C_i, s_i) to the miner.
- Phase 2: t-round anonymization. For $i = 1, \ldots, t$, the miner and the respondents work as follows.
  - At the beginning of the ith round, the miner sends $((C_1, s_1) \ldots, (C_N, s_N))$ to leader i.
  - Leader i checks that she has received N signed messages; if not, she aborts the protocol. Then leader i verifies the signatures: if $i = 1$, for $j = 1, \ldots, N$, she verifies that
  $$V_{y'_j}(C_j, s_j) = \mathsf{accept};$$
  otherwise, for $j = 1, \ldots, N$, she verifies that
  $$V_{y'_{i-1}}(C_j, s_j) = \mathsf{accept};$$
  If any of the above equations does not hold, leader i aborts the protocol.
  - Leader i rerandomizes each piece of data and permutes the pieces: for $j = 1, \ldots, N$,
  $$\begin{aligned} R_j &\stackrel{\mathsf{def}}{=} (R_j^{(1)}, R_j^{(2)}) \\ &= (C_{\pi_i(j)}^{(1)} \cdot y^{\delta_{\pi_i(j)}}, C_{\pi_i(j)}^{(2)} \cdot g^{\delta_{\pi_i(j)}}), \end{aligned}$$
  where π_i is a random permutation on $\{1, \ldots, N\}$, and each δ_j is picked independently and uniformly from $[0, q-1]$.
  - For $j = 1, \ldots, N$, leader i sets $C_j = R_j$ and signs C_j using private key x'_i:
  $$s_j = S_{x'_i}(C_j).$$
  Then leader i sends $((C_1, s_1), \ldots, (C_N, s_N))$ back to the miner.
- Phase 3: Decryption.
  - The miner sends $((C_1, s_1), \ldots, (C_N, s_N))$ to all leaders.
  - Each leader checks that she has received N signed messages; if not, she aborts the protocol. Then she verifies the signatures: for $j = 1, \ldots, N$, she verifies that
  $$V_{y'_t}(C_j, s_j) = \mathsf{accept}.$$
  If any of the above equations does not hold, the leader aborts the protocol.
  - Each leader i computes partial decryptions: for $j = 1, \ldots, N$,
  $$p_{j,i} = (C_j^{(2)})^{x_i}.$$

– Each leader i sends the miner the partial decryptions $(p_{1,i}, \ldots, p_{N,i})$.

– The miner computes the decryptions: for $j = 1, \ldots, N$,

$$d'_j = C_j^{(1)} / \prod_{i=1}^{t} p_{j,i}.$$

4.3 Correctness and Anonymity

The only difference between this protocol and the protocol in the semi-honest model is that her, messages are signed and signatures are verified. Consequently, when all parties follow the protocol, the miner finally obtains a permutation of $(d_1, \ldots, d_N)$.

This protocol preserves the anonymity of each honest respondent against a malicious miner, because if the miner drops or tampers with any message to any respondent, the respondent will detect it (by checking the number of messages and verifying the signatures).

4.4 Efficiency Analysis and Measurements

In this protocol, the computational overhead of a non-leader respondent is 2 modular exponentiations and 1 signing operation. The major computational overhead of a leader is $3N+2$ modular exponentiations, $N+1$ signing operations and $2N+1$ verification operations. The major computational overhead of the miner is Nt modular multiplications and N modular division. The overall communications are at most $(6t+2)\kappa N + (4t+1)\kappa' N$ bits, where κ' is the length of a digital signature, typically 512 or 1024 bits.

We also implemented this protocol and measured the computation times in the environment described in Section 3.6. The digital signature scheme we use is DSA and the length of each signature is 512 bits.

Figure 5 illustrates our measurements of a regular respondent's computation time: it is always about 16ms regardless of the values of N and t. Compared with 15ms for the protocol in the semi-honest model, the increase in computational overhead is minimal.

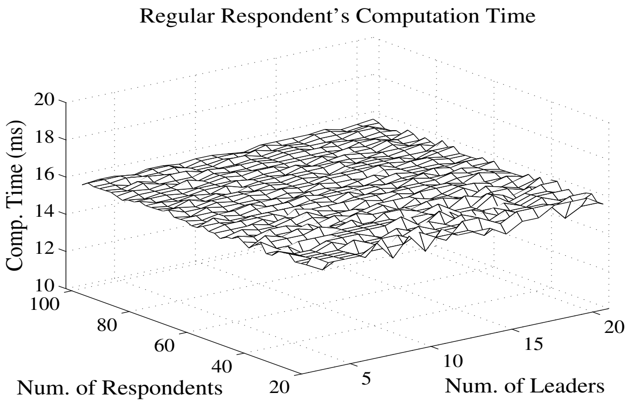

Figure 5: Regular respondent's computation time with malicious miner

Figure 6 illustrates our measurements of a leader's computation time: it is linear in N and does not depend on t. For a typical scenario where $N = 20$, the computation time of a leader is about 0.52s, which has a 10% increase over the corresponding overhead of the protocol in the semi-honest model. Figure 7 illustrates our measurements of the miner's

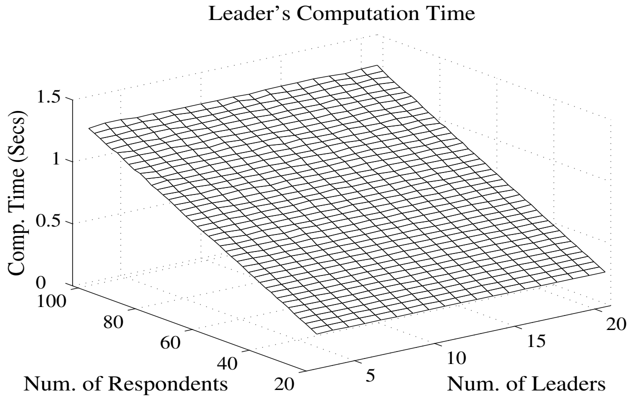

Figure 6: Leader's computation time with malicious miner

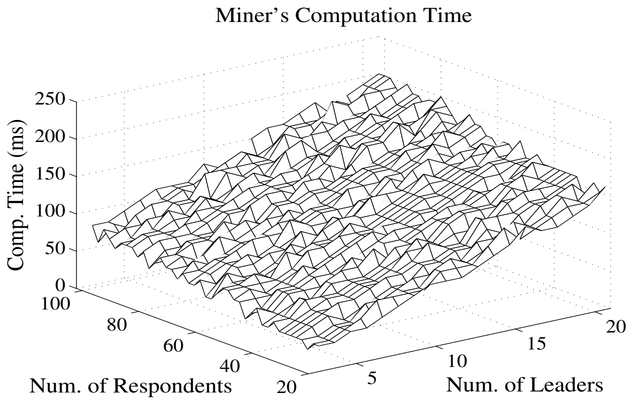

Figure 7: Miner computation time with malicious miner

computation time: it is linear in both N and t. For a typical scenario where $N = 20$ and $t = 3$, the computation time of the miner is about 30ms. For the miner, this protocol against a malicious miner adds no additional computational overhead over the protocol against a semi-honest miner.

5. MALICIOUS MINER & RESPONDENTS

In this section, we consider the case in which some corrupted respondents may also deviate from the protocol. Using the cryptographic tool of *zero-knowledge proofs*, we further extend our solution to work against $t-1$ malicious correspondents in addition to the malicious miner. Before presenting the protocol, we first introduce several types of zero-knowledge proofs it uses.

5.1 Zero-knowledge Proofs

Zero-knowledge proofs (ZKP) are a standard cryptographic tool by which a participant can convince other participants of various statements without leaking any secret information. In this paper, we use three types of ZKP, all of which can be carried out noninteractively (i.e., with only a single message flow):

- $\mathsf{PoK}(C)$, where C is an ElGamal ciphertext. A participant can use this to prove that she knows the cleartext of C.
- $\mathsf{PoR}((C_1, \ldots, C_N), (C'_1, \ldots, C'_N))$, where each C_i and each C'_i are ElGamal ciphertexts. A participant can use this to prove that $(C'_1, \ldots, C'_N)$ is a permuted rerandomization of $(C_1, \ldots, C_N)$, i.e., that $(C'_1, \ldots, C'_N)$ has the same multiset of cleartext messages as $(C_1, \ldots, C_N)$.
- $\mathsf{PoD}(p, C^{(2)}, y)$, where $C^{(2)}$ is the second component of an ElGamal ciphertext and y is a public key. A participant can use this to prove that p is a partial decryption computed by raising $C^{(2)}$ to the private key corresponding to the public key y. Formally, this means $p = (C^{(2)})^x$, where $x = \log y$.

Methods to carry out these proofs can be found in, e.g., [18].

5.2 Protocol

This protocol extends the protocol in the semi-honest model by adding a number of ZKPs. In the data submission phase, each respondent i computes a proof, $z_i = \mathsf{PoK}(C_i)$, proving she knows the cleartext of C_i. Along with C_i, respondent i sends z_i to the miner. The miner forwards (C_i, z_i) to all other respondents. Each respondent verifies the proofs sent by the other $N-1$ respondents. If any proof is missing or invalid, the respondent aborts the protocol.

In the t-round anonymization phase, during round i, leader i generates a proof

$$w_i = \mathsf{PoR}((C_1, \ldots, C_N), (R_1, \ldots, R_N)),$$

which means the new ciphertexts $(R_1, \cdots, R_N)$ are a permuted rerandomization of the old ciphertexts $(C_1, \cdots, C_N)$. When leader i sends the new ciphertexts to the miner, she also sends w_i. The miner forwards them to all other respondents, who verify the proof. If the proof is missing or invalid, then the respondents abort the protocol.

In the decryption phase, each leader i computes a proof

$$v_i = \mathsf{PoD}(p_{j,i}, C_j^{(2)}, y_i),$$

which means $p_{j,i}$ is a partial decryption computed by raising $C_j^{(2)}$ to the private key corresponding to the public key y_i. Each leader i sends the proof v_i along with the partial decryption to the miner, and the miner then forward v_i with partial decryptions to all other respondents. Each respondent verifies the proofs. If any proof is missing or invalid, the protocol is aborted.

In summary, we use ZKP to force the miner and the malicious respondents to follow the protocol. If they do not follow the protocol, their malicious behavior will be detected and the protocol will be aborted.

6. CONCLUSION

In this paper, we propose *anonymity-preserving data collection*, a new approach to protect privacy in data mining. This approach allows a data miner to collect data from a potentially large number of respondents but prevents the miner from finding out which respondent has submitted which piece of data. We present three protocols with provable anonymity guarantees, one in the semi-honest model, one that can tolerate a malicious miner, and one that can tolerate some malicious respondents in addition to a malicious miner. As confirmed by our experiments, the protocols are very efficient.

Our current implementation of the protocols focuses on efficiency measurements, but does not address user interface or architectural issues. In practice, we suggest that the respondent side be implemented as a browser plug-in or a plug-in feature of a web service.

Acknowledgments

We thank the anonymous referees for their helpful comments.

7. REFERENCES

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

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

[3] D. Chaum. Untraceable electronic mail, return address and digital pseudonyms. *Comm. ACM*, 24(2):84-88, 1981.

[4] D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65-75, 1988.

[5] C. Clifton and D. Marks. Security and privacy implications of data mining. In *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 15-19, May 1996.

[6] L. Cranor, editor. *Comm. ACM* 42(2), Special Issue on Internet Privacy, 1999.

[7] T. Dalenius. Finding a needle in a haystack or identifying anonymous census records. *J. Official Statistics*, 2(3):329-336, 1986.

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

[9] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505-510. ACM Press, 2003.

[10] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology - Proceedings of CRYPTO 2003*, volume 3152 of Lecture Notes in Computer Science, Santa Barbara, California, August 2004.

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

[12] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of the Eighth ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, pages 217-228. ACM Press, 2002.

[13] B. Gilburd, A. Schuster, and R. Wolff. k-TTP: a new privacy model for large-scale distributed environments. In *Proc. of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 563-568. ACM Press, 2004.

[14] O. Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[15] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th Annual ACM Conference on Theory of Computing*, pages 218-229. ACM Press, 1987.

[16] S. Goldwasser and S. Micali. Probabilistic encryption. J. Computer and System Sciences, 28:270-299, 1984.

[17] P. Golle and A. Juels. Dining cryptographers revisited. In Advances in Cryptology - EUROCRYPT 2004, volume 3027 of Lecture Notes in Computer Science, pages 456-473, 2004.

[18] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels. Optimistic mixing for exit-polls. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of Lecture Notes in Computer Science, pages 451-465. Springer-Verlag, 2002.

[19] HIPAA. The health insurance portability and accountability act of 1996, October 1998. Available at www.cms.hhs.gov/hipaa.

[20] Markus Jakobsson. Flash mixing. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 83-89. ACM, 1999.

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

[22] B. N. Levine and C. Shields. Hordes - a multicast based protocol for anonymity. *J. Computer Security*, 10(3):213-240, 2002.

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

[24] C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology - Proceedings of EUROCRYPT 93*, volume 765 of LNCS, pages 248-259. Springer-Verlag, 1993.

[25] European Parliament. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official J. European Communities*, page 31, 1995.

[26] European Parliament. Directive 97/66/EC of the European Parliament and of the Council of 15 December 1997 concerning the processing of personal data and the protection of privacy in the telecommunications sector. *Official J. European Communities*, pages 1-8, 1998.

[27] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66-92, 1998.

[28] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proc. of the 28th VLDB Conference*, 2002.

[29] K. Sako and J. Kilian. Receipt-free Mix-type voting schemes — a practical solution to the implementation of a voting booth. In *Advances in Cryptology - Proceedings of EUROCRYPT 95*, volume 921 of Lecture Notes in Computer Science, pages 393-403. Springer-Verlag, 1995.

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

[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] Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In Public Key Cryptography'98, volume 1431 of Lecture Notes in Computer Science, pages 117-134, 1998.

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

[35] Luis von Ahn, Andrew Bortz, and Nicholas J. Hopper. k-anonymous message transmission. In *Proc. 2003 ACM Conference on Computer and Communications Security*, pages 122-130, Washington, DC, 2003.

[36] Michael Waidner. Unconditional sender and recipient untraceability in spite of active attacks. In *EUROCRYPT '89*, pages 302-319, 1989.

[37] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Stat.*, 60(309):63-69, 1965.

[38] R. N. Wright and Z. Yang. Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713-718. ACM Press, 2004.

[39] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proc. of the 2005 SIAM International Conference on Data Mining* (SDM), Newport Beach, California, April 2005.

[40] A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162-167, 1986.