Privacy Preserving Clustering

S. Jha* L. Kruger* P. McDaniel†

Abstract

The freedom and transparency of information flow on the Internet has heightened concerns of privacy. Given a set of data items, clustering algorithms group similar items together. Clustering has many applications, such as customer-behavior analysis, targeted marketing, forensics, and bioinformatics. In this paper, we present the design and analysis of a privacy-preserving k-means clustering algorithm, where only the cluster means at the various steps of the algorithm are revealed to the participating parties. The crucial step in our privacy-preserving k-means is privacy-preserving computation of cluster means. We present two protocols (one based on oblivious polynomial evaluation and the second based on homomorphic encryption) for privacy-preserving computation of cluster means. We have a JAVA implementation of our algorithm. Using our implementation, we have performed a thorough evaluation of our privacy-preserving clustering algorithm on three data sets. Our evaluation demonstrates that privacy-preserving clustering is feasible, i.e., our homomorphic-encryption based algorithm finished clustering a large data set in approximately 66 seconds.

1 Introduction

The ease and transparency of information flow on the Internet has heightened concerns of personal privacy [9, 49]. Web surfing, email, and other services constantly leak information about who we are and what we care about. Many have accepted that some privacy will be lost in exchange for the benefits of digital services [48]. However, in other domains privacy is so important that its protection is federally mandated [1]. Technologies for protecting privacy are emerging in response to these growing concerns [8, 18, 45]. Recently, more emphasis has been placed on preserving the privacy of user-data aggregations, e.g., databases of personal information. Access to these collections is, however, enormously useful. It is from this balance between privacy and utility that the area of *privacy preserving data-mining* emerged [3, 33].

Unsupervised learning deals with designing classifiers from a set of unlabeled samples. A common approach for unsupervised learning is to first cluster or group unlabeled samples into sets of samples that are "similar" to each other. Once the clusters have been constructed, we can design classifiers for each cluster using standard techniques (such as decision-tree learning [38, 44]). Moreover, clusters can also be used to identify features that will be useful for classification. There is significant research on privacy-preserving algorithms for designing classifiers [3, 33]. This paper addresses the problem of privacy-preserving algorithms for clustering.

Assume that Alice A and Bob B have two unlabeled samples D_A and D_B. We assume that each sample in D_A and D_B has all the attributes, or the data sets are horizontally partitioned between A and B. Alice and Bob want to cluster the joint data set $D_A \cup D_B$ without revealing the individual items of their data sets (of course Alice only obtains the clusters corresponding to her data set D_A). In this paper, we assume that

*Computer Sciences Department, University of Wisconsin, Madison, WI.

†Computer Science and Engineering, Pennsylvania State University, University Park, PA.

clustering the joint data set $D_A \cup D_B$ provides better results than individually clustering D_A and D_B. Using a large data set from the networking domain we also demonstrate that clustering the joint data set results in significantly different clusters than individually clustering the data sets (see end of section 5 for details). We present a privacy-preserving version of the k-means algorithm where only the cluster means at the various steps of the algorithm are revealed to Alice and Bob.

There are several applications of clustering [14]. Any application of clustering where there are privacy concerns is a possible candidate for our privacy-preserving clustering algorithm. For example, suppose network traffic is collected at two ISPs, and the two ISPs want to cluster the joint network traffic without revealing their individual traffic data. Our algorithm can be used to obtain joint clusters while respecting the privacy of the network traffic at the two ISPs. An application of clustering to network intrusion detection is presented by Marchette [36]. Clustering has been used for forensics [43] and root-cause analysis for alarms [29]. Clustering has also been used in bioinformatics. For example, Dhillon *et al.* [11] have used clustering to predict gene function. We believe that privacy-preserving clustering can be used in bioinformatics where the data sets are owned by separate organizations, who do not want to reveal their individual data sets.

This paper makes the following contributions:

- We present the design and analysis of privacy-preserving k-means clustering algorithm for horizontally partitioned data (see Section 3). The crucial step in our algorithm is privacy-preserving of cluster means. We present two protocols for privacy-preserving computation of cluster means. The first protocol is based on oblivious polynomial evaluation and the second one on homomorphic encryption. These protocols are described in detail in Section 4.

- We have also have a JAVA implementation of our algorithm. We believe that modular design of our implementation will enable other researchers to use our implementation. Our clustering tool is available by request. We evaluated the two privacy-preserving clustering algorithms on real data sets. Our first conclusion is that privacy-preserving clustering is feasible. For example, for a large data set ($5,687$ samples and 12 features) from the speech recognition domain our homomorphic-encryption-based algorithm took approximately 66 seconds. We also observed that both in bandwidth efficiency and execution overhead algorithms based on homomorphic encryption performed better than the one based on oblivious polynomial evaluation. A detailed discussion of our evaluation is given in Section 5.

2 Related Work

Privacy issues in statistical databases have been thoroughly investigated [2, 10]. Recently privacy-preserving data mining has been a very active area of research. Initial focus in this area was on construction of decision trees from distributed data sets [3, 33]. There is also a significant body of research on privacy-preserving mining of association rules [15, 46, 50]. We will focus on existing work on privacy-preserving clustering.

In general, there are two approaches for designing privacy-preserving machine learning algorithms. The first approach is to use transformations to perturb the data set before the algorithm is applied. This approach for designing privacy-preserving clustering algorithms is taken by several researchers [31, 37, 41]. A second approach to designing privacy preserving algorithms is to use algorithms from the secure-multiparty computation literature. The advantage of this approach over the perturbation approach is that formal guarantees of privacy can be given for these algorithms. This paper takes the latter approach. Vaidya

Algorithm (k-means clustering)
begin initialize $n, c, \mu_1, \cdots, \mu_c$
  do classify n samples according to nearest μ_i, and
    recompute μ_i
    until no change in μ_i's
    return $\mu_1, \mu_2, \cdots, \mu_c$
  end

Figure 1: The k-means clustering algorithm.

and Clifton's [51] work is closest to the one presented in this paper. Vaidya and Clifton present a privacy-preserving k-means algorithm for vertically-partitioned data sets. As already pointed out in the introduction, our paper considers clustering for horizontally-partitioned data. Vaidya and Clifton's algorithm is based on the secure-permutation algorithm of Du and Atallah [13]. However, Vaidya and Clifton's algorithm has to execute Du and Atallah's protocol for every item in the data set. Therefore, their algorithm is not practical for large data sets. Moreover, Vaidya and Clifton did not perform an experimental evaluation of their algorithm. By contrast, the complexity of our algorithm only depends on the number of steps taken by the k-means algorithm and the dimension of the data items. There are distributed clustering algorithms where the goal is to reduce communication costs [12, 30]. These distributed clustering algorithms do not consider privacy. However, it will be interesting to investigate whether these algorithms can be made privacy preserving.

In our implementation, we approximate real numbers using intervals (see appendix C). Finite-precision approximation to functions may leak information. Feigenbaum *et al.* [16] show that approximations to functions can be made private by adding noise.

3 The k-means clustering algorithm

The k-means algorithm [14, 34] is shown in Figure 1. Assume that we are given n samples $x_1, \cdots, x_n$, where each sample is a m-dimensional vector of real numbers. The number of clusters is c. The algorithm maintains c means $\mu_1, \cdots, \mu_c$. Initially, assume that the means are assigned arbitrary values. A sample x_i is deemed to be in the cluster j if it is closest to the mean μ_j, where mean of a cluster $\{x'_1, \cdots, x'_r\}$ is $\frac{x'_1+\cdots,x'_r}{r}$. Distance between two m-dimensional vectors x and y is given by $\sum_{j=1}^{m}(x[j] - y[j])^2$, where $x[j]$ is the j-th element of the vector x. Other distance metrics [14, Chapter 10], such as scatter metrics, can be used instead of the distance metric mentioned above. Each iteration of the k-means algorithms recomputes the means and reclassifies the samples. The algorithm terminates when it detects "no change" in the means. The precise definition of "no change" depends on the specific metric being used. We also assume that the initial cluster means are chosen randomly. There is some research on picking the initial cluster means [4]. Various techniques for picking initial cluster means can be easily incorporated into our algorithm. This issue will not be discussed further in the paper.

3.1 Distributed k-means

Assume that Alice A (party 1) has z samples $\{x_1, \cdots, x_{n_A}\}$, and Bob B (party 2) has $n - n_A$ samples $\{x_{n_A+1}, \cdots, x_n\}$. Each party wants to jointly cluster their samples without revealing any private information. We are assuming that clustering the union of samples from the two parties is more desirable than

clustering the two samples individually.

Assume that there is a trusted third party TTP. A and B perform iterations locally. However, at each iteration the new cluster means μ_is are computed by communicating with the TTP. Let C_i^A and C_i^B be the cluster corresponding to mean μ_i for A and B, respectively. A sends c-pairs $\langle (a_1, b_1), \cdots, (a_c, b_c) \rangle$ to TTP, where $a_i = \sum_{x_j \in C_i^A} x_j$ and $b_i = \mid C_i^A \mid$ (a_i is the sum of samples in cluster C_i^A and b_i is the number of samples in the cluster C_i^A). Analogously, B sends c-pairs $\langle (d_1, e_1), \cdots, (d_c, e_c) \rangle$ to the TTP, where $d_i = \sum_{x_j \in C_i^B} x_j$ and $e_i = \mid C_i^B \mid$. The TTP computes the c means $\langle \mu_1, \cdots, \mu_c \rangle$ and sends them to A and B, where $\mu_i = \frac{a_i + d_i}{b_i + e_i}$. We call this algorithm *distributed k-means* or $D_{k\text{-means}}$.

3.2 Assumptions

Our goal is to design a privacy-preserving k-means that does not use a TTP. Before we present such an algorithm, we state assumptions made in the design of our privacy-preserving algorithm.

Number of parties. In this paper we only present the two party case.

The adversary model. We assume a semi-honest adversary (also called honest but curious adversary model) [20]. There are standard constructions that transform a protocol that is secure in the semi-honest model and produce a protocol that is secure in a more general malicious model (these constructions are called "semi-honest to malicious" compilers, and details of these constructions can be found in [23]).

Information disclosure. Our privacy-preserving algorithm discloses the cluster means at the various steps to the two parties. Therefore, the computation of classifying samples according to the nearest cluster means can be performed locally. Therefore, the complexity of our privacy-preserving algorithm depends only on the number of steps taken by the k-means algorithm and the number of features, but not on the size of the data. This is a desirable property because usually the data sets to be clustered can be very large.

3.3 Privacy-preserving k-means

In order, to create a privacy-preserving version of k-means that does not use a TTP we have to devise a privacy-preserving protocol to compute the cluster means. Consider the computation of a single cluster mean μ_i. Recall that in distributed k-means each party sends (a_i, b_i) and (d_i, e_i) to the TTP, which computes $\frac{a_i + d_i}{b_i + e_i}$; this is precisely the function for which we have to devise a privacy-preserving protocol. This problem can be formally defined as follows:

Definition 1 The *weighted average problem (WAP)* is defined as follows: party 1 has a pair (x, n), where x is a real number and n is a positive integer. Similarly, party 2 has pair (y, m). They want to jointly compute $\frac{x+y}{n+m}$. In other words, we need a privacy-preserving protocol for the following functionality:

$$((x, n), (y, m)) \quad \longmapsto \quad (\frac{x+y}{n+m}, \frac{x+y}{n+m})$$

The notation shown above means that the first and second party provide inputs (x, n) and (y, m) to the protocol and both parties receive output $\frac{x+y}{n+m}$. Notice that WAP is different than the classical problem of computing the averages, where n parties have a number and they jointly want to compute the average without revealing their individual numbers. In the classical problem, the number of parties n is known to all the parties. In WAP, the number of points n and m needs to be kept secret.

Algorithm $PP_{k\text{-means}}$ (privacy-preserving k-means clustering)
begin initialize $n_A, c, \mu_1, \cdots, \mu_c$
  do classify n_A samples according to nearest μ_i
    for $i := 1$ **to** c **step** 1 **do**
      Let C_i^A be the i-th cluster
      Compute $a_i = \sum_{x_j \in C_i^A} x_j$ and $b_i = \mid C_i^A \mid$
      recompute μ_i by invoking the protocol $\mathcal{P}_{WAP}$
    od
    until no change in μ_i
    return $\mu_1, \mu_2, \cdots, \mu_c$
  end

Figure 2: The privacy-preserving k-means clustering algorithm.

Let $\mathcal{P}_{WAP}$ be a privacy-preserving protocol for solving WAP. Two protocols for WAP are presented in Section 4. In the privacy-preserving k-means algorithm (denoted as $PP_{k\text{-means}}$) A and B use $\mathcal{P}_{WAP}$ instead of the trusted third party TTP to compute the cluster means μ_is. The algorithm is shown in Fig 2. We only show the part of the algorithm executing at Alice's (party 1) side. Bob (party 2) will execute a similar algorithm at his side.
Note: Suppose that the initial clusters are picked randomly. For the privacy-preserving algorithm we need a protocol for two parties to jointly pick a common random vector. Such a protocol is called *coin-tossing into the well* and is based on commitment schemes (see [20, Section 7.4.3.1]).

3.4 Proof of Privacy

In this section we provide a proof of privacy for the protocol shown in Figure 2. The proof uses a semi-honest adversary model. Notice that in the distributed k-means algorithm $\mathcal{D}_{k\text{-means}}$ both parties only know their input and output. Definition of privacy is based on the intuition that parties should learn nothing more from the messages used in privacy-preserving protocol, i.e., the messages received by a party during an execution of a privacy-preserving protocol can be "effectively computed" by only knowing its input and output. This idea is formalized below:

Definition 2 Let x and y be inputs of the two parties and $\langle f_1(x,y), f_2(x,y)\rangle$ be the desired functionality, i.e., the first party wants to compute $f_1(x,y)$ and the second wants to compute $f_2(x,y)$. Let Π be a two-party protocol to compute f. The view of the first party after having participated in protocol Π (denoted by $\text{VIEW}_1^{\Pi}(x,y)$) is $(x, r, m_1, \cdots m_t)$, where r are the random bits generated by party 1 and $m_1, \cdots, m_t$ is the sequence of messages received by party 1, while participating in protocol Π. The view $\text{VIEW}_2^{\Pi}(x,y)$ for the second party is defined in an analogous manner.

We say that Π *privately computes* f if there exists probabilistic polynomial-time algorithms (PPTA), denoted by S_1 and S_2 such that

$$\begin{aligned} \{S_1(x, f_1(x,y))\}_{x,y} &\equiv^s \{\text{VIEW}_1^{\Pi}(x,y)\}_{x,y} \\ \{S_2(x, f_2(x,y))\}_{x,y} &\equiv^s \{\text{VIEW}_2^{\Pi}(x,y)\}_{x,y} \end{aligned}$$

In the equation given above, $\equiv^s$ denotes *statistically indistinguishable*. Two probability ensembles $X = \{X_w\}_{w \in S}$ and $Y = \{Y_w\}_{w \in S}$ indexed by S are statistically indistinguishable if for some negligible function $\mu : \aleph \mapsto [0, 1]$ and all $w \in S$,

$$\sum_{\alpha} \mid Pr(X_w = \alpha) - Pr(Y_w = \alpha) \mid \quad < \quad \mu(\mid w \mid)$$

A function $\mu : \aleph \mapsto [0, 1]$ is called *negligible* if for every positive polynomial p, and all sufficiently large n's, $\mu(n) < \frac{1}{p(n)}$. There is a weaker notion of indistinguishability called *computationally indistinguishable*. We will use statistical indistinguishability throughout the paper, but all the results hold even if the weaker notion of indistinguishability is used. Detailed definitions of these concepts can be found in [19, 20].

The privacy-preserving k-means algorithm uses the privacy-preserving protocol $\mathcal{P}_{WAP}$ for the WAP. Assume that the two parties invoke the protocol $\mathcal{P}_{WAP}$ as an oracle, i.e., both parties write their respective inputs (in this case (x, n) and (y, m)) and invoke the oracle which returns the result (in this case $\frac{x+y}{n+m}$). Recall that in the distributed k-means algorithms both parties learn the cluster means at various steps. If we use oracle calls to compute the cluster means, then the two parties also learn only the cluster means. So the views in the two cases are *identical*. Hence, the conditions of definition 2 are trivially satisfied. However, there are additional messages exchanged in the protocol $\mathcal{P}_{WAP}$ used to compute the cluster means. We need to ensure that nothing can be learned from these messages. The privacy of protocol shown in Figure 2 follows from the composition theorem [7] stated below (g is the algorithm shown in Figure 2 and f is the protocol P_{WAP} to solve WAP described in Section 4):

Theorem 1 (Composition Theorem for the semi-honest model): *Suppose that g is privately reducible to f and that there exists a protocol for privately computing f. Then there exists a protocol for privately computing g.*

4 Privacy-Preserving Protocol for the Weighted Average Problem

In the weighted average problem (WAP) we want to find a privacy-preserving protocol for the following functionality:

$$((x, n), (y, m)) \quad \longmapsto \quad (\frac{x+y}{n+m}, \frac{x+y}{n+m})$$

Recall that a protocol for WAP was used in the privacy-preserving k-means algorithm (see Figure 2).

A simple strategy to address this problem is to first approximate the function $\frac{x+y}{n+m}$ by a circuit C, and then use standard constructions [21, 22, 52] to construct a privacy-preserving protocol. Protocols constructed using this strategy have a very high computational overhead. Malkhi *et al.* considered the cost of implementing these protocols in their work in the Fairplay system [35]. They found that the protocol was feasible for small circuits, e.g., a single $\wedge$-gate could be implemented in 410 milliseconds, and more complex integer numerical functions could be implemented on the order of seconds. They further showed the runtimes of these protocols grow quickly with the size of the input and complexity of the implemented function. The most complex function discussed by the authors computed a median of two ten-element integer input sets. This function took over 7 seconds to execute in a LAN environment, and over 16 seconds in an WAN environment. The circuit for computing $\frac{x+y}{n+m}$ is significantly more complex. Hence, with a non-trivial data set,

(Step 1) Party 1 picks a random element $z \in \mathcal{F}$ and computes two new polynomials zP and zQ. In other words, party 1 "blinds" the polynomials P and Q.

(Step 2) Party 2 computes $zP(\alpha)$ and $zQ(\alpha)$ by invoking the protocol for OPE twice, i.e., invokes the protocol $\mathcal{P}_{OPE}(zP, \alpha)$ and $\mathcal{P}_{OPE}(zQ, \beta)$.

(Step 3) Party 2 computes $\frac{P(\alpha)}{Q(\beta)}$ by computing $\frac{zP(\alpha)}{zQ(\beta)}$ and sends it to party 1.

Figure 3: Protocol for PRPE.

a single computation of cluster means may take several minutes to compute. Note that the underlying costs of Fairplay are not artifacts of the design, but simply the cost of implementing the standard protocols; the reported costs were almost completely dominated with circuit setup and the necessary oblivious transfers.

In this section, we present two privacy-preserving protocols for WAP that are more efficient than the standard protocols. The first protocol is based on oblivious polynomial evaluation and the second on homomorphic encryption. Similarity of WAP with a problem that occurs in protocols for generation of shared RSA keys [6, 17] is discussed in appendix B.

4.1 Protocol based on oblivious polynomial evaluation

We will first give a privacy-preserving protocol for a general problem, and then at the end of the subsection demonstrate how we can construct a privacy-preserving protocol for WAP. Consider the following problem.

Definition 3 Let $\mathcal{F}$ be a finite field. Party 1 has two polynomials P and Q with coefficients in $\mathcal{F}$. Party 2 has two points α and β in $\mathcal{F}$. Both parties want to compute $\frac{P(\alpha)}{Q(\beta)}$. In other words, we want to privately compute the following functionality:

$$((P, Q), (\alpha, \beta)) \quad \longmapsto \quad (\frac{P(\alpha)}{Q(\beta)}, \frac{P(\alpha)}{Q(\beta)})$$

We call this problem *private rational polynomial evaluation (PRPE)*.

The protocol $\mathcal{P}_{PRPE}$ uses a protocol for oblivious polynomial evaluation, which is defined below.

Definition 4 Let $\mathcal{F}$ be a finite field. The *oblivious polynomial evaluation* or *OPE* problem can be defined as follows: Alice A has a polynomial P over the finite field $\mathcal{F}$, and Bob B has an element $x \in \mathcal{F}$. After executing the protocol implementing OPE B should *only know* $P(x)$ and A should know nothing.

A protocol to solve the OPE was given by Naor and Pinkas [40]. Let $\mathcal{P}_{OPE}(P, \alpha)$ denote the privacy-preserving protocol for OPE. We provide a protocol $\mathcal{P}_{PRPE}((P, Q), (\alpha, \beta))$ for PRPE, which uses $\mathcal{P}_{OPE}(P, \alpha)$ as an oracle. The protocol is shown in Figure 3.

Theorem 2 Protocol $\mathcal{P}_{PRPE}((P, Q)(\alpha, \beta)$ shown in Figure 3 is privacy-preserving protocol for PRPE.

Proof: The views of the two parties are

$$\begin{aligned} \text{VIEW}_1^{\mathcal{P}_{PRPE}}(P, Q) &= (P, Q, \frac{P(\alpha)}{Q(\beta)}) \\ \text{VIEW}_2^{\mathcal{P}_{PRPE}}(\alpha, \beta) &= (\alpha, \beta, zP(\alpha), zQ(\beta)) \end{aligned}$$

The view of party 1 consists of its input (P, Q) and output $\frac{P(\alpha)}{Q(\beta)}$. Therefore, there is nothing to prove (see definition 2, we can use S_1 as the identity function). The input and output of party 2 are (α, β) and $\frac{P(\alpha)}{Q(\beta)}$ respectively. We have to show a PPTA S_2 such that $S_2(\alpha, \beta, \frac{P(\alpha)}{Q(\beta)})$ and $\text{VIEW}_2^{\mathcal{P}_{PRPE}}(\alpha, \beta)$ are statistically indistinguishable. Let z' be a random element of $\mathcal{F}$ and $S_2(\alpha, \beta, \frac{P(\alpha)}{Q(\beta)})$ be defined as follows:

$$(\alpha, \beta, z'\frac{P(\alpha)}{Q(\beta)}, z')$$

It is easy to see that the following two ensembles are statistically indistinguishable:

$$(\alpha, \beta, z'\tfrac{P(\alpha)}{Q(\beta)}, z')$$
$$(\alpha, \beta, zP(\alpha), zQ(\beta))$$

The reason is that if z is a random element of $\mathcal{F}$ then $zQ(\beta)$ is a random element of $\mathcal{F}$ as well. Moreover, the ratio of the third and fourth elements in the view of party 2 is $\frac{P(\alpha)}{Q(\beta)}$, i.e., the output and the third element of the view determine the fourth element of the view.

Recall that $\mathcal{P}_{PRPE}$ uses the protocol $\mathcal{P}_{OPE}$. Using the composition theorem we conclude that $\mathcal{P}_{PRPE}$ is privacy preserving. □

Protocol for WAP. First, we show that a protocol $\mathcal{P}_{PRPE}$ for PRPE can be used to solve WAP. Recall that in WAP party 1 and party 2 have inputs (x, n) and (y, m) respectively. In the invocation of $\mathcal{P}_{PRPE}$, party 1 constructs two polynomials $P(w) = w + x$ and $Q(w) = w + n$, and party 2 sets $\alpha = y$ and $\beta = m$. The output both parties receive is equal to $\frac{x+y}{n+m}$, which is the desired output. The proof of privacy for this protocol follows from Theorem 2 and the composition theorem.

4.2 Protocol based on homomorphic encryption

Let (G, E, D, M) be a encryption scheme (where G is the function to generate public parameters, E and D are the encryption and decryption functions, and M is the message space respectively) with the following properties:

- The encryption scheme (G, E, D) is *semantically secure* [24]. Essentially, an encryption scheme is semantically secure if an adversary gains no extra information by inspecting the ciphertext. This is formally defined in the appendix (see definition 5).
- For all $m \in M$ and $\alpha \in M$, $m_1 \in E(m)$ implies that $m_1^\alpha \in E(m\alpha)$. Encrypting the same message twice in a probabilistic encryption function can yield a different ciphertext, so $E(m)$ denotes the set of ciphertexts that can be obtained by encrypting m.[1]
- There is a computable function f such that for all messages m_1 and m_2 the following property holds:

$$f(E(m_1), E(m_2)) \quad = \quad E(m_1 + m_2)$$

[1] Of course, to successfully decrypt two different messages m and m' sets $E(m)$ and $E(m')$ should be disjoint.

- **(Step 1)** Party 1 encrypts x and n and sends the encrypted values $x_1 \in E(x)$ and $n_1 \in E(n)$ to party 2.

- **(Step 2)** Party 2 computes a random message $z \in M$, and encrypts $z \cdot y$ and $z \cdot m$ to obtain $z_1 \in E(z \cdot y)$ and $z_2 \in E(z \cdot m)$. Party 2 computes the following two messages and sends it to party 1:

$$\begin{aligned} m_1 &= f(x_1^z, z_1) \\ m_2 &= f(n_1^z, z_2) \end{aligned}$$

  Note: In our implementation we use the homomorphic-encryption scheme by [5] where f is multiplication.

- **(Step 3)** Using the two properties of the probabilistic encryption scheme (G, E, D), we have the following:

$$\begin{aligned} m_1 &= E(z \cdot x + z \cdot y) \\ m_2 &= E(z \cdot n + z \cdot m) \end{aligned}$$

  Therefore, party 1 can compute $z(x + y)$ and $z(n + m)$, and hence can compute $\frac{x+y}{n+m}$. Party 1 sends $\frac{x+y}{n+m}$ to party 2.

Figure 4: Protocol for WAP based on homomorphic encryption.

There are several encryption scheme that have the three properties mentioned above [5, 39, 42]. In our implementation, we used the *dense probabilistic encryption (DPE)* scheme of Benaloh [5]. The semantic security of the scheme provided by Benaloh is based on the intractability of deciding prime residuosity.

Party 1 and 2 have a pair of messages (x, n) and (y, m). The two parties want to jointly compute $\frac{x+y}{n+m}$ in a privacy-preserving way. Assume that party 1 sets up a probabilistic encryption scheme (G, E, D, M), and publishes the public parameters G. We also assume that the probabilistic encryption scheme (G, E, D, M) satisfies the three properties given at the beginning of the section. The protocol $\mathcal{P}_H$ for WAP is shown in Figure 4.

Theorem 3 Assume that the probabilistic encryption scheme (G, E, D) has three properties mentioned at the beginning of this sub-section. $\mathcal{P}_H((x, n), (y, m))$ is a privacy-preserving protocol to compute $\frac{x+y}{n+m}$.

The proof of this theorem is straightforward and is given in appendix A. The basic intuition is that party 2 cannot tell the difference between $E(x)$ and $E(n)$ and encryption of two arbitrary messages.

The complexity of encryption and decryption operations of a scheme (G, E, D, M) depends on size of the message space M. Therefore, in order to keep the complexity low it is important that the size of the message space be small. However, in order to achieve adequate precision the message space should be large. Chinese remainder theorem (CRT) allows us to perform computation over smaller spaces and then reconstruct the result for a larger message space. Let $p_1, \cdots, p_m$ be m small primes. The two parties

execute the protocol described above for $Z_{p_1}, \cdots, Z_{p_m}$. Party 1 receives $z(x+y)$ and $z(n+m)$ modulo p_i (for $1 \leq i \leq m$). CRT allows party 1 to reconstruct $z(x+y)$ and $z(n+m)$ modulo $N = \prod_{i=1}^{m} p_i$. This technique is also used by Gilboa [17].

5 Experimental Evaluation

This section looks at the feasibility of our solution by evaluating the cost of the protocol on real data-sets. The goal of this study is to establish the cost of our privacy-preserving clustering algorithms on real applications. We principally seek to understand the performance and privacy tradeoffs inherent to the operation of the protocols.

We evaluated three clustering algorithms. The *simple* scheme is used throughout as a baseline for our experiments. This protocol implements the k-means clustering algorithm as described in section 3. This algorithm does not use any privacy-preserving protocols. This represents the nominal cost of clustering, and will be present in any k-means clustering approach, independent of if and how privacy is implemented. Throughout this section *features* refer to the dimension of the vectors being clustered and each iteration of the k-means algorithm is referred to as *round*. Our first privacy-preserving protocol (referred to as *OPE*) uses oblivious polynomial evaluation. This protocol is described in detail in Section 4.1. For oblivious polynomial evaluation we use the protocol presented by Naor and Pinkas [40]. The next privacy-preserving protocol (referred to as *DPE*) uses homomorphic encryption scheme of Benaloh [5]. This protocol is described in detail in Section 4.2.

Implementation. Our system consists of approximately 3000 lines of Java code, split up into a number of self-contained modules. The k-means algorithm module implements actual clustering computations as described in Section 3. During each iteration, this module calls the protocol module to compute the cluster means for each dimension of the cluster. The protocol module sets up the framework of communication, and calls the specific protocol handlers with a common interface, depending on which protocol is selected. In the *simple* handler, Alice sends (x, n) to Bob, who computes the cluster mean $\frac{x+y}{n+m}$ and sends it to Alice. The OPE and DPE protocol handlers implement the protocols described in Sections 4.1 and 4.2.

The central results uncovered by this investigation include:

1. Clustering using DPE is two orders of magnitude more bandwidth efficient than OPE, and executes in 4.5 to 5 times less time. This is largely due to bandwidth and computational costs associated with the oblivious transfers used by OPE.
2. Our protocols clustering with perfect fidelity; that is, the clusters resulting from our algorithms are identical to those reported by a k-means algorithm with no privacy for reasonable parameter choices.
3. Small, medium, and large data-sets can be clustered efficiently.
4. Costs scale linearly with feature and rounds. The number of samples affects run-time only inasmuch as it increases the number of rounds toward convergence.
5. Protocol parameters affect bandwidth usage by constant factor. Moreover, exponential increases in security or supported message space result in linear increases in execution run-times.

We begin in the following section by exploring several real data-sets representative of expected environments.

5.1 Experimental Data

The validity of our experimental approach is partially dependent on the realism of our test data. For this reason, we have obtained a collection of externally provided data-sets representing diverse applications. All experiments described in this section use the *synthetic*, *river*, *robot*, and *speech* data-sets detailed below.

We selected the elements of our *synthetic* data-set to enable testing and measure startup costs. This data set includes 4 points uniformly distributed within a 6 dimensional space. By design, the data clusters quickly into 4 "natural" clusters within 2 rounds under the k-means algorithm in all experiments.

Originally used in the Computation Intelligence and Learning (COIL) competition, the *river* data-set describes measurements of river chemical concentrations and algae densities [27]. The river data was used to ascertain the summer algae growth of river water in temperate climates. The clustered data is used to inform the relationship between the presence and concentrations of various chemicals in public waterways and algae growth. The river contains 184 samples with 15 features per sample.

The *robot* data-set [26] contains continuous senor readings from the Pioneer-1 mobile robot used for testing computer learning and conceptual development approaches. Each of the 697 samples contains 36 features from sensor arrays of the Pioneer-1 mobile robot. The samples were taken every 100ms and reflect the movements and changing environment in which the robot was tested. The data has been clustered in prior use to recognize experiences with common outcomes.

The *speech* data-set [28] documents the measured voice characteristics of spoken Japanese vowels. Nine male speakers uttered two Japanese vowels $/ae/$ repeatedly. Sampled at 10kHz, the 640 utterances resulted in 12 features of 5,687 samples. This large data-set is used in the context of our experiments to evaluate the degree to which the proposed protocols scale with the size of the input data. Similar data-sets are clustered frequently to help guide speech recognition software [32].

Each of the data-sets represents a singular corpus. In contrast, our protocols are targeted for applications of clustering with two parties. We model the two party case by randomly subdividing the samples into equal sized subsets and assigning them to each party. In real environments the size of the sets may be vastly different. Our approximation approach ensures that this kind of asymmetry will be transparent to both parties both in execution and performance. That is, the performance of the algorithm is largely independent of the number of samples. However, as we shall see below, the number of features has tremendous effect on the cost of clustering.

The last data set (called the *ping* data-set) was collected by us. The purpose of collecting this data was two fold:

- Test our clustering algorithm on a large data set.
- Construct a data set that can be naturally partitioned to demonstrate that jointly clustering two data sets can produce significantly different results than individually clustering them.

We setup two hosts (referred to as A and B) to measure ICMP ping round-trip times. There were 4 ping targets located around the world (one of the ping targets was on the same subnet as host B). On each host and for each ping target the pings were grouped in blocks of 200. For each block, a 3-tuple consisting of the following three values was generated: the average time to live (TTL), the average round-trip time (RTT), and fraction of lost packets (%drop). We collected data over a period of 24 hours and generated a data set consisting of 23872 data points, which were evenly divided between host A and B. We ran our clustering algorithm on the joint data set, and data sets corresponding to hosts A and B.

5.2 Experimental Setup

We use the architecture and code described earlier for the experiments described throughout. All experiments are executed on a pair of 3Ghz machines with 2 gigabyte physical memory. The experimental application is running on the Sun Microsystems Java Virtual Machine version 1.5 [47] on the Tao Linux version 1.0 operating system [25]. The protocols are executed on a 100Mbps unloaded LAN with a measured round-trip time of 0.2 milliseconds.

The experiments profile the additional cost of providing privacy in clustering sensitive data. To this end, we focus on three metrics of cost and utility; *communication overhead*, *delay*, and *precision*. Communication overhead records the amount of additional network bandwidth used by the privacy schemes over the simple schemes. Delay measures the additional time required to complete the clustering.

Precision is used to measure the degree to which the approximated clustering diverge from those reported by a simple k-means algorithm, and is calculated as follows. Let $X = \{x_1, \ldots, x_n\}$ be the sample data set to be clustered. $C_1 \subseteq 2^X$ is the clustering of X by the simple algorithm, and $C_2 \subseteq 2^X$ is the clustering returned by the OPE algorithm (the DPE metric is defined similarly in the obvious manner). For each pair (x_i, x_j) such that $1 \leq i < j \leq n$ an error occurs if

1. x_i and x_j are in the same cluster in C_1, but in C_2 they are in different clusters.
2. x_i and x_j in the same cluster in C_2, but in C_1 they are in different clusters.

The total number of errors is denoted E. The maximum number of errors is $N = n(n-1)/2$. The precision P is given by $(N - E)/N$.

Both OPE and DPE have unique parameters which dictate the performance and security of each protocol. The performance of DPE is most effected by the size of the primes used to select the homomorphic encryption keys. Small primes can be cryptanalyzed, and large ones can unnecessarily increase bandwidth use and computational costs. Like RSA, linear increases in the size of the primes should result in exponential security improvements.

We use interval arithmetic to approximate real numbers (see appendix C). The size of the message space in DPE and the finite-field in OPE are chosen to achieve the desired precision. In Benaloh's encryption scheme r denotes the size of the message space. For efficiency reasons we choose $r = 3^k$ (see [5] for details). Two crucial parameters in the oblivious polynomial evaluation protocol of Naor and Pinkas are D, the degree of the masking polynomial and M, the total number of points used (details of this algorithm can be found in [40]). The sender's masking polynomial D has degree $k.d$, where d is the degree of the polynomial P being evaluated and k is the security parameter. Since in our algorithm the polynomial being evaluated is always linear, the security parameter is simply D. Increasing D strengthens the sender's security. Only $D + 1$ points are needed to interpolate, but the receiver sends $(D + 1).M$ pairs of values to the sender. Out of each set of M pairs, one of them is related to α (the point the polynomial is being evaluated on), and the other $M - 1$ values are random. The 1-out-of$M$ oblivious transfer protocol (denoted as OT_1^M) is repeated $D + 1$ times to learn the required value. So, increasing M strengthens the receiver's security. Unless otherwise specified, we selected $D = 7$ and $M = 6$. For brevity, we do not consider D or M further.

5.3 Results

Our first battery of tests broadly profile the performance of OPE and DPE. Shown in Table 1, the most striking characteristic of these experiments is that they demonstrate that OPE protocols consume two orders of magnitude more network resources than the DPE protocols. These costs can be directly attributed to

Test	Rounds	Communications Overhead			Delay		
		bytes	bytes feature/rnd	percent increase	milliseconds	milliseconds feature/rnd	percent increase
Synthetic (4 samples, 6 features)							
Simple	2	5959	0	0%	168	0	0%
OPE	2	1497823	124322	25035.48%	10147	831.58	5939.88%
DPE	2	13580	635.08	127.89%	2135	163.9166667	1170.83%
River (184 samples, 15 features)							
Simple	16	74574	0	0%	772	0	0%
OPE	16	29916457	124241.17	40116.47%	176133	730.67	22715.16%
DPE	16	234422	566.03	314.35%	38721	158.12	4915.67%
Robot (697 samples, 36 features)							
Simple	8	94005	0	0%	1348	0	0%
OPE	8	36569040	126649.42	38801.16%	212776	734.125	15684.57%
DPE	8	269698	610.04	186.90%	47662	160.8125	3435.76%
Speech (5,687 samples, 12 features)							
Simple	33	143479	0	0%	4198	0	0%
OPE	33	49359739	124183.48	34402.07%	294694	733.57	6919.87%
DPE	33	384644	509.00	268.08%	66101	156.3207071	1474.58%
Ping (28,392 samples, 3 features)							
Simple	9	11644	0	0%	2765	0	0%
OPE	9	3429688	126594.2	29354.55%	23767	777.8519	759.566%
DPE	9	30633	703.29	163.07%	9694	256.63	250.59%

Table 1: Experimental Results - resource and precision results from experiments over the three data sets. The feature/round statistics show the costs of per feature clustering in a single round of the k-means algorithm, e.g., a single execution of the privacy preserving WAP protocol.

the oblivious transfer algorithms whose primitive cryptographic operations require the transfer of many polynomials between hosts. The total bandwidth costs scaled linearly for both OPE and DPE. That is, the bandwidth costs per feature/round are relatively constant for the given data sets, where we observed 0.03% variance in scaled bandwidth usage in OPE and 9.36% in DPE. Note that the bandwidth is ultimately of limited interest; the worst case experiment only consumes 47 megabytes of bandwidth over two and a half minutes. Hence, our protocols would have visible impact only the slowest or busiest networks.

A chief feature illustrated by the timing measurements is that DPE is much more time and bandwidth efficient than OPE. Surprisingly, DPE is 4.5 to 5 times faster on all the data-sets for the selected parameters. The reasons for this is that the underlying oblivious transfers incur large message exchanges between the two parties. Hence, in all experiments the limiting factors are bandwidth and computation.[2] The efficiency of DPE with respect to OPE further shows fixed costs (startup) are likewise dominated by the underlying privacy preservation operations. Further, like the bandwidth costs, the execution of each algorithm scale linearly with the number of features and rounds, where each feature round requires 730 and 160 milliseconds for OPE and DPE to complete, respectively.

The cost of privacy-preservation in large data-set clustering is noticeable. For example, a large data-set containing 5687 samples and 12 features takes DPE just 66 seconds to cluster, as opposed to the 4.19 seconds required by its simple k-means counterpart. Hence for this experiment, DPE algorithm incurs slowdown of a factor of 15 and the more expensive OPE a factor of 70. These results are, for most applications, clearly within the bounds of acceptable performance. This is particularly encouraging in the face of past attempts; circuit implementations of vastly simpler operations (averaging very small collections of data points) took tens of seconds to complete [35].

For the parameters we selected the precision of our privacy-preserving algorithms (DPE and OPE) was 100%. The reasons for this are two-fold. The parameter choices for DPE resulted in a message space of 3^{40} values, which allowed us to map cluster means to 4 decimal places. Moreover, the data range was small in all our data-sets. Hence, the error rounding caused by using interval arithmetic was inconsequential. Note that in other environments, where the message space is required to be smaller (likely for performance reasons) or the range of data values is large, precision errors may be introduced.

The costs of OPE grow slightly with increases in D and M. We experimented with varied parameters of D and M equal 5, 10, 15 on all the non-synthetic data-sets (for a total of 27 experiments) . In all cases increased cost was nominal; the parameter sets slowed the performance of the algorithm down between 60% and 190% over a baseline experiment, i.e., $M = D = 5$. Again, these costs are a direct reflection of the costs of the underlying oblivious transfer. Not shown, the bandwidth costs in DPE scale by a constant factor proportional to D and M.

As illustrated in Figure 5, increases the size n (which is a product of two primes) in DPE has modest affect on the performance of the protocols. Exponential increases in n result in linear increases in message size. Because the network is a limiting factor, such increases are, as shown, reflected in linear slowdowns. Hence, very large intervals or high precision clustering can be supported by small increases in bandwidth consumption. As in OPE, bandwidth costs in DPE scale by a constant factor in these experiments, where each protocol exchange increases directly in proportion to the size of the primes.

For the ping data set our clustering algorithm generated 4 clusters, which correspond to the four target hosts. The centers for the four clusters are shown in Figure 6. As can be clearly seen from the results, clusters found by the algorithm using the joint data set are significantly different than the clusters found in

[2]Early implementations of our protocols were limited by the latency caused by many individual round-trips in the protocol. We optimized these these by parallelizing exchanges, where possible. This vastly improved protocol performance, and as a direct result, bandwidth and and computation have since emerged as the limiting factors.

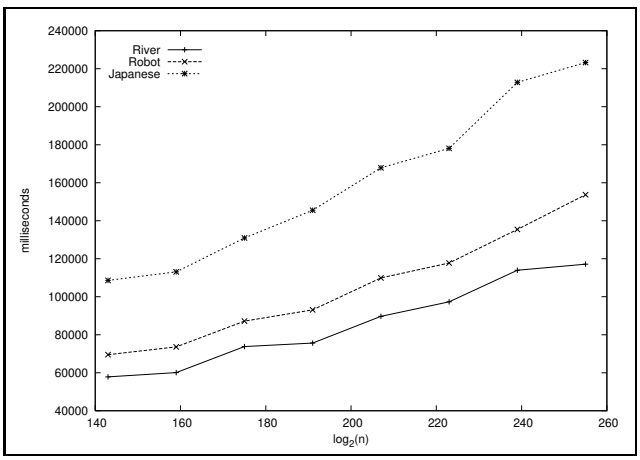

Figure 5: DPE runtime costs by message space - in *milliseconds*, the time to cluster the sample data-sets with various widths of n message spaces.

	Cluster centers
A	$(241.76, 32.69, 0.18), (48.00, 75.87, 0.58), (243.00, 59.81, 0.15), (64.00, 0.19, 0.00)$
B	$(47.00, 88.60, 0.74), (251.92, 4.73, 0.19), (242.00, 48.01, 2.70), (133.67, 485.77, 13.78)$
Joint	$(245.26, 28.73, 0.60), (47.51, 82.13, 0.66), (133.67, 485.77, 13.78), (64.00, 0.186, 0.00)$

Figure 6: (TTL,RTT,%drop) centers for the four clusters.

the individual data sets. Therefore, if the goal is to estimate RTT, TTL, and %drop for the target hosts to be used in networking applications (such as routing), then clustering on the joint data set is desirable.

6 Conclusion

We presented two privacy-preserving k-means algorithms. We also implemented these algorithm and performed a thorough evaluations of our algorithms. There are several avenues for further research. We want to perform further optimizations to our tool to reduce the execution and bandwidth overheads. We want to explore privacy-preserving versions of other clustering algorithms. We are particularly interested in hierarchical clustering algorithms.

Acknowledgments

We thank Dan Boneh for pointing out the connection between the weighted average problem and generation of shared RSA keys.

References

[1] 104th Congress. *Public Law 104-191: Health Insurance Portability and Accountability Act of 1996*, August 1996.

[2] N.R. Adam and J.C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21, 1989.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, TX, May 2000.

[4] P.S. Bardley and U.M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of 15th International Conference on Machine Learning (ICML)*, pages 91–99, 1998.

[5] J. Benaloh. Dense probabilistic encryption. In *Workshop on Selected Areas of Cryptography*, pages 120–128, May 1994.

[6] D. Boneh and M. K. Franklin. Efficient generation of shared RSA keys. *Journal of the ACM (JACM)*, 48(4):702–722, 2001.

[7] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[8] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. W3C Recommendation, 16 April 2002.

[9] Lorrie Faith Cranor. Internet privacy. *Communications of the ACM*, 42(2):28–38, 1999.

[10] D.E. Denning. A security model for the statistical database problem. *ACM Transactions on Database Systems (TODS)*, 5, 1980.

[11] I.S. Dhillon, E.M. Marcotte, and U. Roshan. Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, 19(13):1612–1619, 2003.

[12] I.S. Dhillon and D.S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Proceedings of Large-scale Parallel KDD Systems Workshop (ACM SIGKDD)*, August 15-18 1999.

[13] W. Du and M. J. Atallah. Privacy-preserving cooperative statistical analysis. In *Annual Computer Security Applications Conference ACSAC)*, pages 102–110, New Orleans, Louisiana, USA, December 10-14 2001.

[14] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[15] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, Edmonton, Alberta, Canada, July 23–26 2002.

[16] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In *28th International Colloquium Automata, Languages and Programming (ICALP 2001)*, Crete, Greece, July 8-12 2001.

[17] Niv Gilboa. Two party rsa key generation. In *Advances in Cryptology (CRYPTO '99)*, Santa Barbara, California, USA, August 15-19 1999.

[18] Ian Goldberg, David Wagner, and Eric Brewer. Privacy-enhancing technologies for the internet. In *Proc. of 42nd IEEE Spring COMPCON*. IEEE Computer Society Press, February 1997.

[19] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, May 2001.

[20] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[21] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th Symposium on Theory of Computer Science*, pages 218–229, 1987.

[22] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[23] O. Goldreich and E. Petrank. Quantifying knowledge complexity. In *Computational Complexity*, volume 8, pages 50–98, 1999.

[24] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Science*, 28:270–299, 1984.

[25] Tao Linux User Group. Tao Linux, version 1.0. http://taolinux.org/, November 2004.

[26] Information and Computer Science. *Pioneer-1 Mobile Robot Data*. University of California Irvine, November 1998. `http://kdd.ics.uci.edu/databases/pioneer/pioneer.html`.

[27] Information and Computer Science. *COIL 1999 Competition Data, The UCI KDD Archive*. University of California Irvine, October 1999. `http://kdd.ics.uci.edu/databases/coil/coil.html`.

[28] Information and Computer Science. *Japanese Vowels*. University of California Irvine, June 2000. `http://kdd.ics.uci.edu/databases/JapaneseVowels/JapaneseVowels.html`.

[29] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):443–471, November 2003.

[30] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):405–421, November 2001.

[31] M. Klusch, S. Lodi, and Gianluca Moro. Distributed clustering based on sampling local density estimates. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 485–490, 2003.

[32] M. Kudo, J. Toyama, and M. Shimbo. Multidimensional Curve Classification Using Passing-Through Regions. *Pattern Recognition Letters*, (11–13):1103–1111, 1999.

[33] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology (Crypto 2000)*, pages 36–54, August 2000.

[34] S.P. Llyod. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, IT-2:129–137, 1982.

[35] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay – A Secure Two-Party Computation System. In *Proceedings of 13th USENIX Security Symposium*, pages 287–302. USENIX, September 2004. San Diego, CA.

[36] D. Marchette. A statistical method for profiling network traffic. In *Workshop on Intrusion Detection and Network Monitoring*, pages 119–128, 1999.

[37] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 211–218, 2003.

[38] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[39] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS)*, San Francisco, California, 1998.

[40] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *31st Symposium on Theory of Computer Science*, pages 245–254, Atlanta, GA, May 1-4 1999.

[41] S. Oliveira and O. R. Zaiane. Privacy preserving clustering by data transformation. In *XVIII Simpósio Brasileiro de Bancos de Dados, 6-8 de Outubro (SBBD 2003)*, pages 304–318, 2003.

[42] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology (EUROCRYPT'99)*, 1999.

[43] F. Pouget and M. Dacier. Honeypot-based forensics. In *Proceedings Of AusCERT Asia Pacific Information technology Security Conference 2004(AusCERT2004)*, Brisbane, Australia, May 2004.

[44] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[45] D. M. Rind, I. S. Kohane, P. Szolovits, C. Safran, H. C. Chueh, and G. O. Barnett. Maintaining the confidentiality of medical records shared over the internet and the world wide web. *Annals of Internal Medicine*, 127(2), July 1997.

[46] S.J. Rizvi and J.R. Harista. Maintaining data privacy in association rule mining. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, Hong Kong, August 20-23 2002.

[47] Sun Microsystems. Sun Java Virutal Machine, version 1.5. http://java.sun.com/, November 2004.

[48] Humphrey Taylor. Most people are "privacy pragmatists" who, while concerned about privacy, will sometimes trade it off for other benefits. *The Harris Poll*, (17), March 19 2003.

[49] Joseph Turow. Americans and online privacy: The system is broken. Technical report, Annenberg Public Policy Center, June 2003.

[50] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, Edmonton, Alberta, Canada, July 23–26 2002.

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

[52] A.C. Yao. How to generate and exchange secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

A Definitions and Proofs

Definition 5 Assume that the message space M can be sampled in polynomial time, i.e., there exists a probabilistic polynomial time algorithm A_M such that it takes input 1^k and generates a message $m \in M$. Let $h : M \rightarrow R$ be a function, which can be thought of as some information about the message, e.g., $h(m) = 1$ iff message has a substring "Bob" in it. Consider the following two games:

- **(Game 1):** Adversary is informed that I am about to choose a message m using the sampling algorithm A_M. The adversary is asked to guess $h(m)$.
- **(Game 2):** In addition to the information given in game 1, he is also told the encryption $\alpha \in E(m)$ of the message. The adversary is again asked to guess $h(m)$.

An encryption function E is called *semantically secure* if the difference between the probabilities of the adversary succeeding in the two games is negligible. The probability is computed over the message space.

Proof of Theorem 3: The view of the two parties is shown below:

$$\begin{aligned} \text{VIEW}_1^{\mathcal{P}_H}(x,n) &= (x, n, z(x+y), z(n+m)) \\ \text{VIEW}_2^{\mathcal{P}_H}(y,m) &= (y, m, x_1, n_1, \frac{x+y}{n+m}) \end{aligned}$$

Let z' be a message uniformly chosen from M. Define $S_1(x, n, \frac{x+y}{n+m})$ as follows:

$$(x, n, z' \frac{x+y}{n+m}, z')$$

It is easy to see that $S_1(x, n, \frac{x+y}{n+m})$ and $\text{VIEW}_1^{\mathcal{P}_H}(x,n)$ are statistically indistinguishable (this proof is very similar to the proof of Theorem 2 given in Section 4.1).

Recall that $x_1 \in E(x)$ and $n_1 \in E(n)$. Since (G, E, D) is semantically secure, party 2 cannot gain extra information from the encrypted values x_1 and n_1. In other words. Let $x_1' \in E(x')$ and $n_1' \in E(n')$, where x' and n' are randomly chosen messages. An adversary cannot distinguish between $\text{VIEW}_2^{\mathcal{P}_H}(y,m)$ and $(y, m, x_1', n_1', \frac{x+y}{n+m})$ with more than negligible probability. Therefore, privacy of party 1 with respect to party 2 follows. □

B Generation of shared RSA keys and WAP

We assume that all elements are drawn from a finite field $\mathcal{F}$. Suppose that party 1 and 2 have a pair of numbers (a, b) and (c, d) and they want to privately compute $(a + c)(b + d)$. In other words, they want to privately compute the following functionality:

$$((a, b), (c, d)) \longmapsto (a + c)(b + d)$$

This problem is one of the crucial steps in the protocol for sharing RSA keys. Let $\mathcal{P}_{sk}$ be the protocol for solving this problem. We will show that $\mathcal{P}_{sk}$ can be used to design a protocol $\mathcal{P}_{WAP}$ for solving WAP (see Section 4 for a description of this problem). Protocol $\mathcal{P}_{WAP}$ works as follows:

- Party 1 and party 2 generate two random elements z_1 and z_2 chosen uniformly from $\mathcal{F}$.

- Two parties invoke the protocol $\mathcal{P}_{sk}$ with inputs (x, z_1) and (y, z_2). Each party obtains $r_1 = (x + y)(z_1 + z_2)$.
- Two parties invoke the protocol $\mathcal{P}_{sk}$ with inputs (n, z_1) and (m, z_2). Each party obtains $r_2 = (n + m)(z_1 + z_2)$.
- The two parties obtain $\frac{x+y}{n+m}$ by computing $\frac{r_1}{r_2}$.

Next we argue that $\mathcal{P}_{WAP}$ is privacy preserving. The views of the two parties in this protocol are:

$$\begin{aligned} \text{VIEW}_1(x, n) &= (x, n, (x + y)(z_1 + z_2), (n + m)(z_1 + z_2)) \\ \text{VIEW}_2(y, m) &= (y, m, (x + y)(z_1 + z_2), (n + m)(z_1 + z_2)) \end{aligned}$$

Let z' be a random element of $\mathcal{F}$ and $S_1(x, n, \frac{x+y}{n+m})$ be defined as follows:

$$(x, n, z' \frac{x + y}{n + m}, z')$$

If we fix x, y, and z_1 and pick z_2 uniformly from $\mathcal{F}$, then $(x + y)(z_1 + z_2)$ is a random element distributed uniformly over $\mathcal{F}$. Therefore, $\text{VIEW}_1(x, n)$ and $S_1(x, n, \frac{x+y}{n+m}))$ are statistically indistinguishable. Let z' be a random element of $\mathcal{F}$ and $S_2(y, m, \frac{x+y}{n+m}))$ be defined as follows:

$$(y, m, z' \frac{x + y}{n + m}, z')$$

It is easy to see that $\text{VIEW}_2(y, m)$ and $S_2(y, m, \frac{x+y}{n+m}))$ are statistically indistinguishable. Using the composition theorem the privacy of $\mathcal{P}_{WAP}$ follows.

C Approximating Reals

Assume that real numbers occur in the interval $[M, -M)$. We divide the interval $[M, -M)$ into $2MN$ sub-intervals of size $\frac{1}{N}$. The i-th sub-interval (where $0 \leq i < 2MN$) is given by

$$\left[-M + \frac{i}{N}, -M + \frac{i + 1}{N}\right)$$

We denote by $I(x)$ as the sub-interval the real number x lies in, i.e. $x \in [-M + \frac{I(x)}{N}, -M + \frac{I(x)+1}{N})$. If x and y are two real numbers that lie in the sub-interval $I(x)$ and $I(y)$, then $x + y$ lies in the sub-interval $[-2M + \frac{I(x)+I(y)}{N}, -2M + \frac{I(x)+I(y)+2}{N})$.

For the rest of the sub-section we will approximate real numbers with the the interval they lie in. In our protocol, a party obtains $z(I(x) + I(y))$ and $z(n + m)$, where z is the random number. Using some simple arithmetic we can deduce that $\frac{z(I(x)+I(y))}{z(n+m)}$ lies in the interval $[-M + \frac{Q}{N}, -M + \frac{Q+1}{N})$, where Q is the quotient of q_1 divided by q_2. Integers q_1 and q_2 are shown below:

$$\begin{aligned} q_1 &= MN(z(n + m) - 2) + z(n + m) \cdot z(I(x) + I(y)) \\ q_2 &= z(n + m) \end{aligned}$$

In all our algorithms, we have to use a large enough space so that all the operations used to calculate q_1 and q_2 are exact, i.e., there is no "wrap around". If all the integers used in q_1 and q_2 are bounded by 2^k, then the size of the field should be greater than or equal to 2^{4k+5}.