Assuring Privacy when Big Brother is Watching

Murat Kantarcıoĝlu   Chris Clifton
Department of Computer Sciences
Purdue University
250 N University St
West Lafayette, IN 47907-2066
{kanmurat, clifton}@cs.purdue.edu

ABSTRACT

Homeland security measures are increasing the amount of data collected, processed and mined. At the same time, owners of the data raised legitimate concern about their privacy and potential abuses of the data. Privacy-preserving data mining techniques enable learning models without violating privacy. This paper addresses a complementary problem: What if we want to *apply* a model without revealing it? This paper presents a method to apply classification rules without revealing either the data or the rules. In addition, the rules can be verified not to use "forbidden" criteria.

Keywords

Privacy, Security, Profiling

1. INTRODUCTION

The new U.S. government CAPPS II initiative plans to classify each airline passenger with a green, yellow or red risk level. Passengers classified as green will be subject to only normal checks, while yellow will get extra screening and red won't fly. [2] Although government agencies promise that no discriminatory rules will be used in the classification and that privacy of the data will be maintained, this does not satisfy privacy advocates.

One solution is to have the government send the classification rules to the owners of the data (e.g., credit reporting agencies). The data owners would then verify that the rules are not discriminatory, and return the classification for each passenger. The problem with this approach is that revealing the classification rules gives an advantage to terrorists. For example, knowing that there is a rule "A one-way ticket paid in cash implies yellow risk", no real terrorist will buy one way tickets with cash.

This appears to give three conflicting privacy/security requirements. The data must not be revealed, the classifier must not be revealed, and the classifier must be checked for validity. Although these seem contradictory, we show that if such a system *must* exist, it can be done while achieving significant levels of privacy. We prove that under reasonable assumptions (i.e., the existence of one way functions, non-colluding parties) it is possible to do classification similar to the above example that provably satisfies the following conditions:

- No one learns the classification result other than designated party.
- No information other than the classification result will be revealed to designated party.
- Rules used for classification can be checked for the presence of certain conditions without revealing the rules.

While such a system still raises privacy issues (particularly for anyone classified as red), the security risks are significantly reduced relative to a "give all information to the government" (or anyone else) model.

2. RELATED WORK

This work has many similarities with privacy-preserving distributed data mining. This was first addressed for the construction of decision trees[11]. This was based on the concepts of secure multiparty computation (discussed below.) There has since been work to address association rules in horizontally partitioned data[6; 7], EM Clustering in Horizontally Partitioned Data[10], association rules in vertically partitioned data[13], and generalized approaches to reducing the number of "on-line" parties required for computation[8]. However, the goal in this paper is not to *learn* a data mining model, but to privately *use* a model. (The other approach to privacy-preserving data mining, adding noise to data before the mining process(i.e [1]), doesn't make sense when the goal is to privately evaluate the results of the model on specific data items.) A good survey of privacy and data mining can be found in [9].

2.1 Secure Multiparty Computation

In the late 1980's, work in secure multiparty computation demonstrated that a wide class of functions can be computed securely under reasonable assumptions (see Theorem 2.1.) We give a brief overview of this work, concentrating on material that is used later in the paper. For simplicity, we concentrate on the two party case. Extending the definitions to the multiparty case is straightforward.

Secure multiparty computation has generally concentrated on two models of security. The semi-honest model assumes each party follows the rules of the protocol, but is free to later use what it sees during execution of the protocol to compromise security. The malicious model assumes parties can arbitrarily "cheat", and such cheating will not compromise either security or the results (i.e., the results from the nonmalicious parties will be correct, or the malicious party will be detected.)

We assume an intermediate model: preserving privacy with non-colluding parties. A malicious party may corrupt the results, but will not be able to learn the private data of other parties without colluding with another party. This is a realistic assumption for our problem: Both parties want to get correct results (class of a passenger), and are unwilling to compromise this goal through malicious behavior. The real security issue is the potential for misuse of the data or rules if accidentally released. The risk that a terrorist would discover rules stored at an airline or credit agency is much greater than the risk that they would alter the complex software to defeat the system, particularly as altering the protocol to maliciously learn the rules could be detected by inserting passengers with known security classes as a continuous test of the system.

The formal definition of secure multiparty computation[5] provides a methodology for proving the security of our approach. Informally, this definition states that a computation is secure if the view of each party during the execution of the protocol can be effectively simulated by the input and the output of the party. This is not quite the same as saying that private information is protected. For example, assume two parties use a secure protocol to compare two positive integers. If A has 2 as its integer and the comparison result indicates that 2 is bigger than equal other site's integer, A can conclude that B has 1 as its input. Site A can deduce this information by solely looking at its local input and the final result - the disclosure is a fault of the problem being solved and not the protocol used to solve it. To prove a protocol is secure, we have to show that anything seen is not giving more information then seeing the final result.

Yao introduced secure multiparty computation with his millionaire's problem (and solution). Two millionaires want to know who is richer, without either disclosing their net worth[14]. Goldreich extended this to show there is a secure solution for *any* functionality[5]. The general secure two party evaluation is based on expressing the function $f(x, y)$ as a circuit and encrypting the gates for secure evaluation. This leads to the following theorem:

THEOREM 2.1. *[4] Suppose that trapdoor permutation exist. Then any circuit is privately computable (in the semi-honest model).*

Trapdoor permutations can be constructed under the intractability of factoring assumption.

This works as follows. The function f to be computed is first represented as a combinatorial circuit. Each party sends a random bit to the other party for each input, and replaces their input with the exclusive or of their input and the random bit. This gives each party a share of each input, without revealing anything about the input. The parties then run a cryptographic protocol to learn their share of the result of each gate in the circuit. At the end, the parties combine their shares to obtain the final result. This protocol has been proven to produce the desired result without disclosing anything except that result.

3. PRIVATE CONTROLLED CLASSIFICATION

We first formally define the problem of classifying items using decision rules, when no party is allowed to know both the rules and the data, and the rules must be checked for "forbidden" tests. This problem could be solved using the generic method described above. While we have a construction and proof of such a circuit, it is omitted due to space constraints. Instead, we present a comparatively efficient approach based on the notion of an untrusted third party that processes streams of data from the data and rule sources.

3.1 Problem Statement

Given an instance x from site Data with v attributes, we want to classify x according to a rule set R provided by site Government. Let us assume that each attribute of x has n bits, and let x_i denotes the i^{th} attribute of x. We assume that each given classification rule $r \in R$ is of the form $(L_1 \wedge L_2 \wedge \cdots \wedge L_v) \rightarrow C$ where C is the predicted class if $(L_1 \wedge L_2 \wedge \cdots \wedge L_v)$ evaluates to true. Each L_i is either $x_i = a$, or a don't care (always true). (While the don't care clauses are redundant in the problem definition, they will need to be included explicitly in the protocol to mask the number of clauses in each rule. By using don't cares, G can define rules with an arbitrary number of clauses; the other parties gain no information about the number of "real" clauses in the rule.) We assume that for any given x only one rule will be satisfied.

In addition, D has a set F of rules that are not allowed to be used for classification. In other words, D requires $F \cap R = \emptyset$. The goal is to find the class value of x according to R while satisfying the following conditions:

- D will not be able to learn any rules in R,
- D will be convinced that $F \cap R = \emptyset$ holds, and
- G will only learn the class value of x and what is implied by the class value.

3.2 Untrusted Non-colluding Site

To achieve a solution that is both secure and efficient, we make use of an untrusted, non-colluding site. Use of such a site was first suggested in [3]. Application to privacy-preserving data mining was discussed in [8]. The key to such a site is that without active help from one of the other sites it learns nothing, although by colluding with other sites it may be able to obtain information that should not be revealed. Thus, the only trust placed in the site is that it will not collude with any of the other sites to violate privacy. Although this seems like a strong assumption, it occurs often in real life. For example, bidders or sellers on e-bay assume that e-bay is not colluding with other bidders or sellers against them. In our approach, the untrusted site learns only the number of literals v, the number of rules $|R|$, and how many literals of a given rule are satisfied. It does not learn what those literals are, what the class is, how they relate to literals satisfied by other rules or other data items, or anything else except what is explicitly stated above.

3.3 Commutative Encryption

A key tool used in this protocol is commutative encryption. An encryption algorithm is commutative if the following two equations hold for any given feasible encryption keys $K_1, \ldots, K_n \in K$, any item to be encrypted $m \in M$, and any permutations of i, j: $\forall m_1, m_2 \in M$ such that $m_1 \neq m_2$

$$E_{K_{i_1}}(\dots E_{K_{i_n}}(m)\dots) = E_{K_{j_1}}(\dots E_{K_{j_n}}(m)\dots) \quad (1)$$

and for any given $k, \epsilon < \frac{1}{2^k}$

$$Pr(E_{K_{i_1}}(\dots E_{K_{i_n}}(m_1)\dots) = E_{K_{j_1}}(\dots E_{K_{j_n}}(m_2)\dots)) < \epsilon \quad (2)$$

Commutative encryption is used to check if two items are equal without revealing them. For example, assume that party A has item i_A and party B has item i_B. To check if the items are equal, each party encrypts its item and sends it to the other party: Party A sends $E_{K_A}(i_A)$ to B and party B sends $E_{K_B}(i_B)$ to A. Each party encrypts the received item with its own key, giving party A $E_{K_A}(E_{K_B}(i_B))$ and party B $E_{K_B}(E_{K_A}(i_A))$. At this point, they can compare the encrypted data. If the original items are the same, equation 1 ensures that they have the same encrypted value. If they are different, equation 2 ensures that with high probability the encrypted values are different. During this comparison, each site sees only the other site's encrypted value. Assuming the security of the encryption, this simple scheme is a secure way to check equality.

One example of a commutative encryption scheme is Pohlig-Hellman [12], based on the assumption of the intractability of the discrete logarithm problem. This or any other commutative encryption scheme will work for our purposes.

3.4 Protocol for Private Controlled Classification

We now show how to solve the problem presented in Section 3.1 between sites D, G, and an untrusted, non-colluding site C, where C learns only the number of attributes v, the number of rules $|R|$, and the number of literals satisfied by each rule for a given instance x.

The basic idea behind the protcol is that sites D and G send synchronized streams of encrypted data and rule clauses to site C. The order of attributes are scrambled in a way known to D and G, but not C. Each attribute is given two values, one corresponding to don't care, the other to its true value. Each clause also has two values for each attribute. One is simply an "invalid" value (masking the real value). The other is the desired result, either the a (for a clause $x_j = a$), or the agreed upon "don't care" value. C compares to see if either the first or second values match, if so then either the attribute is a match or the clause is a don't care. If there is a match for every clause in a rule, then the rule is true.

The key is that the don't care, true, and invalid values are encrypted differently for each data/rule pair in the stream, in a way shared by D and G but unknown to C. The order (is the first attribute the value, or the don't care value) also changes, again in a way known only to D and G. Since all values are encrypted (again, with a key unknown to C), the non-colluding site C learns nothing except which rule matches. Since the rule identifier is also encrypted, this is useless to C.

The protocol operates in three phases: encryption, prediction, and verification. Three methods are used for encryption, a one-time pad based on a pseudo-random number generator shared by D and G, a standard encryption method E to encrypt data sent to C, and a commutative method Ec for checking for forbidden rules. To aid in understanding the discussion, a summary of the functions / symbols used is given in Table 1.

Table 1: Function and Symbol Descriptions

Symbol	Description
E_K	Encryption with key K
Ec_K	Commutative encryption with key K
R_g	Common pseudo-random generator shared by site D, G
x	Data item to be evaluated, a vector of attributes x_j
A	Rules × attributes matrix of encrypted instances created by site D
$R[i]$	Rule i, consisting of clauses corresponding to attributes of x
B	Rules × attributes matrix of encrypted rules created by site G
$n_j, (n_j + 1)$	Values outside the domain of j^{th} attribute
i	Index for rules
j	Index for attributes
σ	Index for "match" and "invalid" pairs

Protocol 1 Private classification: encryption phase

```
Require: D and G share a pseudo-random generator $R_g$,
  G has a private generator $Rpg$. $R_g(k)$ ($Rp_g(k)$) represents
  the $k^{th}$ number generated by the pseudo-random generators. Let $n$ be a vector of elements where $n_j$ and $n_j + 1$
  are values out side the domain of $x_j$.
  $cntd = cntg = cntgp = 0$ ;

  At site D:
  $K_r = R_g(cntd++)$;
  for $i = 1; i \leq |R|; i++$ do
    for each attribute $x_j$ of $x$ do
      $\sigma = (R_g(cntd++) \bmod 2)$
      $A[i][j][\sigma] = E_{K_r}(x_j \oplus R_g(cntd++))$ ;
      $A[i][j][(1-\sigma) \bmod 2] = E_{K_r}(n_j \oplus R_g(cntd++))$ ;
    end for
  end for
  send $A$ to site $C$

  At site G:
  randomly permute rules in $R$;
  $K_r = R_g(cntg++)$;
  let $R[i][j]$ is $i^{th}$ rule's $j^{th}$ literal;
  let $R[i][v+1]$ is the predicted class for $i^{th}$ rule;
  for $i = 1; i \leq |R|; i++$ do
    for $j = 1; j \leq v; j++$ do
      $\sigma = (R_g(cntg++) \bmod 2)$
      if $R[i][j]$ is of the form $X_j = a_j$ then
        $B[i][j][\sigma] = E_{K_r}(a_j \oplus R_g(cntg++))$ ;
        $B[i][j][(1-\sigma) \bmod 2] = E_{K_r}((n_j+1) \oplus R_g(cntg++))$ ;
      else
        $B[i][j][\sigma] = E_{K_r}((n_j+1) \oplus R_g(cntg++))$ ;
        $B[i][j][(1-\sigma) \bmod 2] = E_{K_r}(n_j \oplus R_g(cntg++))$ ;
      end if
    end for
    $r_c = Rp_g(cntpg++)$;
    $\bar{B}[i] = (r_c, E_{k_r}(r_c) \oplus R[i][v+1])$;
  end for
  send $B, \bar{B}$ to site $C$;
```

In the encryption phase, site D first encrypts every attribute of x using xor with a random number. The xor with random is effectively a one-time pad to prevent revealing anything to site C. An additional attribute is added corresponding to the don't care condition using the "illegal" value n_j not in the domain of x_j. Site G creates encrypted values for each literal based on whether it must match or is a don't care. For the don't care $B[i][j][(1-\sigma) \bmod 2]$ will be the same as $A[i][j][(1-\sigma) \bmod 2]$ (n_j xored with the same random number). G applies a slightly different cryptographic scheme for class values (again padding with a newly generated random each time), ensuring C doesn't see that different rules may have the same class value. This is necessary to ensure C doesn't learn about class distribution of different instances over multiple runs of the protocol. This is repeated for every rule (giving multiple encryptions of the same x, but with different encryption each time.) The encryption phase is describe in detail in Protocol 1.

Protocol 2 Private classification: prediction phase

```
Require: $A, B, \bar{B}$ be generated and sent to site $C$.
  At site C:
  for $i = 1; i \leq |R|; i++$ do
    if $\forall j, 1 \leq j \leq v, (A[i][j][0] == B[i][j][0] \vee A[i][j][1] == B[i][j][1])$ then
      $(rs, cs) = \bar{B}[i]$;
      break;
    end if
  end for
  randomly generate $r_f$
  send $(rs, cs \oplus r_f)$ to D and send $r_f$ to G;

  At site D:
  receive $(rd, cd)$ from site $C$;
  send $cd \oplus E_{k_r}(rd)$ to site $G$

  At site G:
  receive $r$ from site $C$ and $c$ from site $D$
  output $c \oplus r$ as the classification result
```

Site C compares the vectors to find which rule is satisfied in its entirety. However, it cannot directly send the prediction result to site G as this would reveal which rule is satisfied, since this is the encrypted (and distinct for each rule) value rather than the actual class. C instead sends the result xored with a random number to site D. D decrypts this to get the class, but the true class value is masked by the random generated by C. Finally G can combine the information it gets from site C and site D to learn the classification. This process is fully described in Protocol 2.

3.5 Checking for Forbidden Rules

Protocols 1 and 2 generate the correct classification without revealing rules or data. Protocol 3 shows how C and D can test if $F \cap R = \emptyset$ (presumably before D returns the masked result to G in Protocol 2.)

The main idea of Protocol 3 is that equality can be checked without revealing the items using commutative encryption. Since site D knows which random numbers are used by site G, it can mask F in the same way using those random numbers and encrypt with E_{K_r}. Site D also gets the masked R after encryption by site C, encrypts those, and returns them to C. C now has the double encrypted version of the

Protocol 3 Private classification: verification phase

```
Require: Let $\bar{B}$ be the rule set received from $G$ in Protocol 1, $Ec$ be a commutative encryption method.
  At site C:
  $K = \{$ create $2 * |F| * v$ random keys $\}$
  for $i = 1; i \leq |F|; i++$ do
    for $j = 1; j \leq v; j++$ do
      $En[i][j][0] = Ec_{K[i][j][0]}(B[r_f][j][0])$
      $En[i][j][1] = Ec_{K[i][j][1]}(B[r_f][j][1])$
    end for
  end for
  send $En$ to site $D$.

  At site D:
  receive $En$;
  $Kd = \{$ create $2 * |F| * v$ random keys $\}$
  for $i = 1; i \leq |F|; i++$ do
    for $j = 1; j \leq v; j++$ do
      $\bar{En}[i][j][0] = Ec_{Kd[i][j][0]}(En[i][j][0])$
      $\bar{En}[i][j][1] = Ec_{Kd[i][j][1]}(En[i][j][1])$
    end for
  end for
  Generate $F_e$, the matrix of encrypted forbidden rules, from $F$ using the method used by $G$ to generate $B$ from $R$ in Protocol 1.
  for $i = 1; i \leq |F|; i{+}{+}$ do
    for $j = 1; j \leq v; j++$ do
      $Ef[i][j][0] = Ec_{Kd[i][j][0]}(F_e[i][j][0])$
      $Ef[i][j][1] = Ec_{Kd[i][j][1]}(F_e[i][j][1])$
    end for
  end for
  send $Ef$,$\bar{En}$ to site $C$

  At site C:
  receive $Ef$, $\bar{En}$;
  for $i = 1; i \leq |F|; i++$ do
    for $j = 1; j \leq v; j++$ do
      $Ez[i][j][0] = Ec_{K[i][j][0]}(Ef[i][j][0])$
      $Ez[i][j][1] = Ec_{K[i][j][1]}(Ef[i][j][1])$
    end for
  end for
  for $i = 1; i \leq |F|; i++$ do
    if $\forall j, 1 \leq j \leq v, (\bar{En}[i][j][0] == Ez[i][j][0] \vee \bar{En}[i][j][1] == Ez[i][j][1])$ then
      output that $F \cap R = \emptyset$ does not hold.
    end if
  end for
```

classification rule set R. It now encrypts F. The items can be compared since $Ec_{k_1}(Ec_{k_2}(A)) = Ec_{k_2}(Ec_{k_1}(A))$.

3.6 Security of the Protocol

To prove that this protocol reveals nothing but the number of matching literals, we use the secure multiparty computation paradigm of defining a simulator whose output is computationally indistinguishable from the view seen by each party during execution of the protocol. This reduces to showing that the received messages can be simulated; the algorithm itself generates the rest of the view. During the encryption phase, D and G receive no messages. Assuming encryption is secure, output of the encryption is computationally indistinguishable from a randomly chosen string over the domain of the encryption output. We can define the simulator as follows: The simulator randomly generates numbers and assigns them to A_s. For B_s, first a number of locations corresponding to the number of matching literals are chosen at random. For these locations, $A[i][j][\sigma]$ is used for $B[i][j][\sigma]$. For other locations, random values are generated.

Assume that the output of the simulator's A_s, B_s is **not** computationally indistinguishable from the A, B seen during the protocol. Let M be the distinguisher. Then M must be able to distinguish between some $A[i][j][\sigma]$ and $A_s[i][j][\sigma]$. This means M can distinguish between a random number and $E_{K_r}(X \oplus R)$, contradicting our secure encryption assumption. For $\bar{B}$ a similar argument applies.

For the classification phase site D only sees r, d. Since both of them are random numbers, a simple random number generator can be used for the simulator. The message G receives can be simulated using a random number generator (simulating the message from C) and the actual result ($result \oplus r$).

For the verification phase note that what each party sees is an item encrypted by a random key for each field. Assume that the places of the literals that are equal is the part of the final result (changing the order of literals at each execution prevents this from revealing any real information.) Site C can use the same simulator given for the encryption phase. Therefore we can conclude that the method is secure under the stated assumptions.

In practice the site C would operate in a stream mode, with D sending each new instance x and G re-encrypting the rules R. To C these would appear to be continuous streams – the repetition in R is masked by the random pad. This avoids C learning anything based on results over multiple instances (e.g., giving C a single encrypted rule set for all instances would enable C to learn what rule was most common, even if it didn't know what that rule was.)

One interesting advantage to this method over the generic method is that by colluding (e.g., under a court order), any two sites could reveal what the third site has. For example if G does use a forbidden rule, C and D can collaborate to expose what that rule is. This is not directly possible with generic circuit evaluation, since a malfeasant G could delete the keys used in the protocol to permanently hide the forbidden rule.

4. COST ANALYSIS

Privacy is not free. Keeping the necessary information private requires many encryptions for each classification. Given v literals in each rule both sites G and D perform $O(|R| \cdot v)$ encryptions in the encryption phase. The prediction phase requires a single encryption. Verification requires $O(|F| \cdot v)$ encryptions. The total number of encryptions is then $O((|R| + |F|) \cdot v)$.

The communication cost of the protocol is similar. Assume the size of each encrypted value is t bits. The encryption phase sends $O(|R| \cdot v \cdot t)$ bits, and the prediction phase sends $3t$ bits. In the verification phase the set F is transmitted in encrypted form, $O(|F| \cdot v \cdot t)$ bits. The total communication cost is $O((|R| + |F|) \cdot v \cdot t)$ bits.

While the communication cost may seem high, particularly since this is the cost per instance to be evaluated, it is likely to work well in practice. Bandwidth is growing rapidly, it is generally latency that limits performance. This protocol adapts well to streaming - the small number of messages, and the fact that each site sends only one per phase, means a continuous stream of instances could be fed through the system. The throughput of the system could approach the bandwidth of the communication network.

Note that the generic circuit evaluation is likely to be significantly more expensive. We have a circuit construction that solves the problem with $O((|R| + |F|) \cdot v \cdot n))$ encryptions where n is the number of bits to represent a literal. The circuit size is $O(|R| \cdot |F| \cdot v \cdot n)$ gates, giving a bit transmission cost of $O(|R| \cdot |F| \cdot v \cdot n \cdot t)$. Due to the necessity of representing all possible input in the construction of the circuit, a significantly more efficient approach based on generic circuit evaluation is unlikely.

5. CONCLUSIONS

As the usage of data mining results for homeland security and other potentially intrusive purposes increases, privately *using* these results will become more important. We have shown that it is possible to ensure privacy without compromising the ultimate goal.

The protocol presented here is a first cut at addressing this problem. Developing a practical solution requires additional work. One example is negative clauses. We have a method to easily extend this protocol to handle clauses with negation, however it reveals at least an upper bound on the number clauses with negation. Developing a more secure approach, and proving that secure, is an open topic. Other issues include non-boolean rules (e.g., best match), allowing multiple rule matches, supporting less structured data formats (e.g., text), and demonstrating practical efficiency.

Continued research in privacy preserving data mining must take into account not just generating the results, but efficiently and privately using them. Another research direction is lower bounds on computation and the communication cost. We would like to explore the trades off between cost and privacy. We believe that research can develop methods to increase security in our lives while sacrificing less privacy than generally assumed.

6. REFERENCES

[1] 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 14-19 2000. ACM.

[2] Computer assisted passenger pre-screening II program.

http://www.fcw.com/fcw/articles/2003/0310/news-tsa-03-10-03.asp.

[3] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *26th ACM Symposium on the Theory of Computing (STOC)*, pages 554–563, 1994.

[4] O. Goldreich. Secure multi-party computation, Sept. 1998. (working draft).

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

[6] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, pages 24–31, Madison, Wisconsin, June 2 2002.

[7] M. Kantarcıoĝlu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE-TKDE*, submitted.

[8] M. Kantarcioglu and J. Vaidya. An architecture for privacy-preserving mining of client information. In C. Clifton and V. Estivill-Castro, editors, *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, volume 14, pages 37–42, Maebashi City, Japan, Dec. 9 2002. Australian Computer Society.

[9] Special section on privacy and security. *SIGKDD Explorations*, 4(2):i–48, Jan. 2003.

[10] X. Lin and C. Clifton. Privacy preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, Submitted.

[11] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54. Springer-Verlag, Aug. 20-24 2000.

[12] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24:106–110, 1978.

[13] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 23-26 2002.

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