Secure multiparty computation

Gagan Aggarwal

November 6, 2003

1 Introduction

Two parties, Alice and Bob, hold inputs A and B respectively. They wish to compute $f(A,B)$ without revealing any more information about their respective inputs to the other party. Ideally, we would have liked to give the input to a trusted third party, which can compute the function $f(A,B)$ and give the output to both Alice and Bob. In the absence of such a trusted third party, Alice and Bob would like to engage in a *secure* protocol at the end of which they learn as much information as they would have in the third party scenario. A protocol is called secure if given its own input and the output, each of the parties can produce a transcript which is computationally indistinguishable from the transcript of the real protocol. In this sense, the parties learn no more information than that learnt from just the output.

2 Generic Solution

Yao gave a generic construction which does the following. Given any function which can be computed by a circuit, one can design a protocol by which the two parties are able to compute the function securely.

For each wire W in the circuit, Alice chooses two values W_0 and W_1 (called the garbled values of the wire) corresponding to the boolean values 0 and 1 which the wire can take in any evaluation. In addition, for each gate in the circuit, she encodes the truth table of the gate in terms of the garbled values of the input and output wires. She gives the garbled values corresponding to her input wires to Bob; i.e. if W is her input wire, and her input for that wire is 0, then she gives the value W_0 to Bob, else if her input is 1, she gives the value W_1 to Bob. To transfer the garbled values corresponding to Bob's input wires, Alice and Bob engage in *Oblivious transfer* (see Mayank's writeup for more information). Given these garbled inputs, Bob needs to evaluate the circuit in terms of garbled values. For this, Alice provides Bob with the garbled truth tables in an encrypted form. The entry corresponding to garbled inputs W_X, and W_Y are encrypted with a key derived from W_X and W_Y. Thus, given a pair of garbled input values for a gate, Bob can decrypt the entry corresponding to that input and compute the garbled output. In this manner, he proceeds to compute the garbled output of the circuit, which he sends back to Alice, who does the decoding and sends the output to Bob. Since Alice cannot see the intermediate values, she learns nothing more about Bob's input other than the output. Although Bob can see the intermediate values, he cannot interpret them in terms of zeros and ones and hence learns nothing from them.

2.1 Primitive used: Oblivious transfer

Problem: Alice has two number x_0 and x_1, and Bob has a bit b.
Goal: Bob learns x_b but not x_{1-b}. Alice learns nothing.
Solution: described by Mayank.

2.2 Overhead

This generic protocol requires an oblivious transfer for each input bit. The computation overhead (in terms of number of exponentiations required) is linear in the size of the circuit. It might become prohibitively expensive for database applications where the number of input values are typically very large.

3 Database problems

Input: Alice holds database D_A, Bob holds database D_B.

3.1 Problems looked at

1. Find the intersection/equijoin of D_A with D_B. [Agrawal, Evfimievski, Srikant]

   Agrawal, Evfimievski, Srikant. *Information sharing across private databases.* http://portal.acm.org/citation.cfm?doid=872757.872771

   Communication overhead is linear in the size of the databases.

2. Find the intersection size/join size of D_A with D_B. Notice that this is not necessarily easier than doing (1) above, since we are not allowed to reveal any extra information. [Agrawal, Evfimievski, Srikant]

   Communication overhead is linear in the size of the databases.

3. Find the intersection $D_A \cap D_B$, and intersection size $|D_A \cap D_B|$. Same as above problem. Solved for the malicious case too (when users may not stick to the specified protocol and behave arbitrarily) Pinkas, Wirth (?). Manuscript.

4. Association Rule Mining in Vertically Partitioned Data. [Clifton, Vaidya]

   C. Clifton, J. Vaidya. *Privacy Preserving Association Rule Mining in Vertically Partitioned Data.*
   www.cs.purdue.edu/homes/clifton/DistDM/kdd02.pdf

This paper doesn't stick to the strict definition of privacy and reveals more information than just the output. e.g. it reveals the value of some linear combinations of inputs.

5. Association Rule Mining in Horizontally Partitioned Data. [Kantarcioglu, Clifton]

   M. Kantarcioglu, C. Clifton. *Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data (2002).*
   http://www.cs.purdue.edu/homes/clifton/document/Kantarcioglu.pdf

   This paper solves this problem under the strict privacy definition given above.

6. k-center, k-median [Aggarwal, Mishra, Pinkas]

   Manuscript.

3.2 Open Problems

1. Suitable definition of privacy for NP-hard problems (since trusted third party cannot also produce the exact answer, so what should privacy be defined with respect to?) Maybe define privacy with respect to the Trusted third party running a particular algorithm.

2. Facility location on $D_A \cup D_B$.

3. More efficient computation of the intersection size. Current results require linear communication overhead in the size of the database. Finding even an approximation with the sublinear overhead is open.

4. Another related problem is finding the number of distinct elements in $D_A \cup D_B$. If we can find intersection exactly, then this problems also get solved exactly. However, the two problems behave differently in terms of approximations. In this context, look at "Secure Multiparty Computation of Approximations" by Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin Strauss, and Rebecca Wright. This paper solves the hamming distance problem between two databases (number of uncommon elements) approximately with sublinear overhead, but does not imply a subliear protocol for approximating the intersection size.