Security Enhanced Privacy Preserving Data sharing With Random ID Generation

Authors
Ayswarya R Kurup¹, Simi I²

¹M.Tech Student: Department of Computer Science and Engineering
Musaliar College of Engineering and Technology, Kerala, India.
Email-ajishkcprojects@gmail.com, mob: 9400913778
²Assistant Professor: Department of Computer Science and Engineering
Musaliar College of Engineering and Technology, Kerala, India.
Email-simiindiradevi@gmail.com, mob: 9747903140

ABSTRACT
Need for sharing of private data with security is a major concern in personal and business use of internet. An algorithm for sharing of private data with enhanced security from databases among users is developed. The user can access the databases of other members of the group without knowing to each other. This technique is used iteratively to assign these users random generated ID numbers ranging from 1 to N. These IDs are unknown to other members of this group. This ID is encrypted using some cryptographic algorithms in order to share the databases. This assignment of serial numbers allows more complex data to be shared and has applications to other problems in privacy preserving data mining, collision avoidance in communications and distributed database access. This algorithm can be used in a distributed computing environment.

Existing and new algorithms for assigning random IDs are examined with respect to trade-offs between communication and computational requirements. The new algorithms are built on top of a secure sum data mining operation using DES algorithm for applying encryption. Application of encryption made access of database more secure. An algorithm for distributed solution of certain polynomials over finite fields enhances the scalability of the algorithms.

Index Terms—Distributed computing systems, secure multiparty computation, privacy preserving data mining, privacy protection, security in group communication.
INTRODUCTION

We consider a scenario where a group of users having private databases wish to cooperate by computing a data mining algorithm on the union of their databases. But they can preserve the privacy of their identity and the contents of their databases. Data mining is a recently emerging field, connecting the three worlds of databases, artificial intelligence and statistics. The information age has enabled many organizations to gather large volumes of data. If the meaningful information cannot be extracted data becomes ineligible. Data mining attempts to discover this need. A key problem arises in large collection of data is that of confidentiality. The need for privacy can be motivated by business interests.

The popularity of internet as a communication medium whether for personal or business use depends in part on its support for anonymous communication. Businesses also have legitimate reasons to engage in anonymous communication and avoid the consequences of identity revelation. For example, to allow dissemination of summary data without revealing the identity of the entity the underlying data is associated with, or to protect whistle-blower’s right to be anonymous and free from political or economic retributions [1].

Researchers have also investigated the relevance of anonymity and/or privacy in various application domains: patient medical records, electronic voting, e-mail, social networking, etc.

Another form of anonymity, as used in secure multiparty computation, allows multiple parties on a network to jointly carry out a global computation that depends on data from each party while the data held by each party remains unknown to the other parties. A secure computation function widely used in the literature is secure sum that allows parties to compute the sum of their individual inputs without disclosing the inputs to one another. Therefore there exists a secure protocol for any probabilistic polynomial time functionality. This secure sum is popular in data mining applications and also helps characterize the complexities of the secure multiparty computation.

It is clear that any reasonable solution may have the individual parties do the majority of the computation independently. Our solution is based on this guiding principle and in fact, the number of bits communicated is dependent on the number of transactions by a logarithmic factor only. The necessary condition for obtaining such an algorithm is the existence of a distributed algorithm with low communication complexity.

This work deals with efficient algorithms for assigning identifiers (IDs) to the nodes of a network in such a way that the IDs are anonymous that is randomly generated using a distributed computation with no central authority. Given $N$ nodes, this assignment is essentially a permutation of the integers $\{1, \ldots, N\}$ with each ID being known only to the node to which it is assigned. Our main algorithm is based on a method for anonymously sharing simple data and results in methods for efficient sharing of complex data. Such IDs can be used as part of schemes for sharing/dividing communications bandwidth, data storage, and other resources anonymously and without conflict. The IDs are needed in sensor networks for security or for...
administrative tasks requiring reliability, such as configuration and monitoring of individual nodes, and download of binary code or data aggregation descriptions to these nodes. An application where IDs need to be anonymous is grid computing where one may seek services without divulging the identity of the service requestor.

To elaborate this consider a situation where N parties wish to display their data collectively, but anonymously, in N slots on a third party site. The IDs can be used to assign the N slots to users, while anonymous communication can allow the N parties to conceal their identities from the third party. In another application, it is possible to use secure sum to allow one to opt-out of a computation beforehand on the basis of certain rules in statistical disclosure limitation or during a computation and even to do so in an anonymous manner.

The work reported in this paper further explores the connection between sharing secrets in an anonymous manner, distributed secure multiparty computation and random ID assignment. Our network is not anonymous and the participants are identifiable in that they are known to and can be addressed by the others. This paper builds an algorithm for sharing simple integer data on top of secure sum and DES algorithm. The sharing algorithm will be used at each iteration of the algorithm for random ID assignment (RIDA). This RIDA algorithm, and the variants that we discuss, can require a variable and unbounded number of iterations. Increasing a parameter in the algorithm will reduce the number of expected rounds. However, our central algorithm requires solving a polynomial with coefficients taken from a finite field of integers modulo a prime. That task restricts the level to which can be practically raised.

1.1 Related Work

Secure two party communication was first investigated by Yao and later generalized to multiparty computation. These works all use a similar methodology: the functionality f to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. While this approach is appealing in its generality and simplicity, the protocols it generates depends on the size of the circuit. This size depends on the size of the input, and on the complexity of expressing f as a circuit. We stress that secure two-party computation of small circuits with small inputs may be practical using the protocol.

2. SECURE SUM

Should all pairs of nodes have a secure communication channel available, a simple, but resource intensive, secure sum algorithm can be constructed. In the following algorithm, it is useful to interpret the values as being integer on first reading
Table 1: Random Numbers Transmitted By A Secure Sum Execution

<table>
<thead>
<tr>
<th>Nodes</th>
<th>( r_{i,1} )</th>
<th>( r_{i,1} )</th>
<th>( r_{i,2} )</th>
<th>( r_{i,3} )</th>
<th>( r_{i,4} )</th>
<th>( d_i )</th>
<th>( \hat{d}_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{i=1} )</td>
<td>13 - 6 + 8 = 15</td>
<td>13 - 10</td>
<td>6</td>
<td>-3</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>( n_{i=2} )</td>
<td>7 - 10 + 9 = 6</td>
<td>7</td>
<td>3</td>
<td>-5</td>
<td>5</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>( n_{i=3} )</td>
<td>-8 - 6 + 5 = -9</td>
<td>-8</td>
<td>11</td>
<td>12</td>
<td>-9</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>( n_{i=4} )</td>
<td>6</td>
<td>6</td>
<td>-8</td>
<td>-5</td>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Algorithm 1 (Secure Sum): Given nodes \( n_1, ..., n_N \) each holding an data item \( d_i \) from a finitely representable abelian group, share the value \( T = \sum d_i \) among the nodes without revealing the values \( d_i \).

Each node \( n_i, i=1...N \) chooses random \( r_{i,1}, ..., r_{i,N} \) values such that

\[ r_{i,1} + ... + r_{i,N} = d_i \]

Each “random” value \( r_{i,j} \) is transmitted from node \( n_i \) to node \( n_j \). The sum of all these random numbers \( r_{i,j} \) is, of course, the desired total \( T \).

Each node \( n_j \) totals all the random values received as:

\[ S_j = r_{i,1} + ... + r_{i,N} \]

Now each node \( n_i \) simply broadcasts \( S_i \) to all other nodes so that each node can compute:

\[ T = S_1 + ... + S_N \]

Example 1 (A Secure Sum Computation): In Table 1 examples are shown. In the example, the initial data items held by nodes \( n_1, n_2, n_3 \) and \( n_4 \) are, \( d_1 = 6, d_2 = 10, d_3 = 6 \) and \( d_4 = 2 \) respectively. For example, node \( n_2 \) would transmit 7, 3, -5, and 5 to nodes \( n_1, n_2, n_3 \), and \( n_4 \) respectively. Node \( n_2 \) would receive 3, 11, and -8 from nodes \( n_1, n_2, n_3 \), and \( n_4 \), respectively. Then node \( n_2 \) would compute and transmit the total \( S_2 = -4 \) of the values received to all nodes. Finally, \( n_2 \) would compute the total of all the second round transmissions received, 24 = 18 + 4 + 8 + 2.

3. RIDA (Random ID Assignment Algorithm)

Given nodes \( n_1, ..., n_N \) each holding a data item \( d_i \) from a finitely representable field \( F \), make their data items public to all nodes without revealing their sources.

1) Each node \( n_i \) computes \( d_i^n \) over the field \( F \) for \( n = 1, ..., N \). The nodes then use secure sum to share knowledge of the power sums:

\[ P_1 = \sum_{i=1}^{10} d_i^1 \]
\[ P_2 = \sum_{i=1}^{10} d_i^2 \]
\[ P_N = \sum_{i=1}^{10} d_i^N \]
2) The power sums $P_1, \ldots, P_N$ are used to generate a polynomial which has $d_1, \ldots, d_N$ as its roots using Newton’s Identities. Representing the Newton polynomial as 

$$p(x)=c_Nx^N + \ldots + c_1x + c_0 \quad C_0, \ldots, c_N$$

are obtained from the equations 

$$c_N = -1$$

$$c_{N-1} = -\frac{1}{1}(c_N P_1)$$

$$c_{N-2} = -\frac{1}{2}(c_{N-1} P_1 + c_N P_2)$$

3) The polynomial is solved by each node, or by a computation $p(x)$ distributed among the nodes, to determine the roots $d_1, \ldots, d_N$.

The power sums $p_i$ can be collected and shared using a single round of secure sum.

### 4. ALGORITHM TO FIND RIDA

Given nodes $n_1, \ldots, n_N$, use distributed computation (without central authority) to find an anonymous indexing permutation.

$$s: \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$$

1) Set the number of assigned nodes $A=0$.

2) Each unassigned node $n_i$ chooses a random number $r_i$ in the range 1 to $s$. A node assigned in a previous round chooses $r_i=0$.

3) The random numbers are shared anonymously. Denote the shared values by $q_1, \ldots, q_N$.

4) Let $q_1, \ldots, q_k$ denote a revised list of shared values with duplicated and zero values entirely removed where $k$ is the number of unique random values. The nodes $n_i$ which drew unique random numbers then determine their index $s_i$ from the position of their random number.

5) Update the number of nodes assigned: $A=A+k$.

6) If $A<N$ then return to step(2).

### 5. FINITE TERMINATION

Although the algorithms developed here terminate with probability 1, there is no absolute upper bound on the number of rounds required. Under some assumptions, it has been proven that finite termination cannot be guaranteed for the simpler leader election problem. While there may be extreme conditions under which no algorithm for AIDA can be guaranteed to finitely terminate, we conjecture only that at least $N$ sequential communications are required in such an algorithm. On the other hand, the algorithms of are already collision free, but do not generate a permutation chosen at random from all possible permutations.

For the current problem, the number of rounds is typically small and we do not recommend seeking finitely bounded termination. For completeness, we sketch a cryptographic approach, that could guarantee finitely bounded termination, even without a trusted authority. Suppose each node $n_i$ has a unique, but not
anonymous identifier, \(1 \leq A_i \leq N\). Then the numbers are unique where is \(R_i = E(s_iN + A_i)\) an encryption function, and is the usual sufficiently long seed (random number) known only to . The function may be cooperatively generated with inverse unknown to the nodes individually using techniques from. The use of these random numbers at step (2) of Algorithm would guarantee termination in a single round. However, polynomial solution for of the required size is impractical.

6. CONCLUSION

Each algorithm is compared and its advantages are evaluated. Use of the Newton identities greatly decreases communication overhead. This can enable the use of a larger number of “slots” with a consequent reduction in the number of rounds required. The solution of a polynomial can be avoided at some expense by using Sturm’s theorem. The development of a result similar to the Sturm’s method over a finite field is an enticing possibility.

With private communication channels, these algorithms are secure in an information theoretic sense. Apparently, this property is very fragile. The very similar problem of mental poker was shown to have no such solution with two players and three cards. The argument of can easily be extended to, e.g., two sets each of \(N\) colluding players with a deck of \(2N+1\) cards rather than our deck of \(2N\) cards.

All of the noncryptographic algorithms have been extensively simulated, and we can say that the present work does offer a basis upon which implementations can be constructed. The communications requirements of the algorithms depend heavily on the underlying implementation of the chosen secure sum algorithm. In some cases, merging the two layers could result in reduced overhead.

REFERENCES


