Technical Report

TR-2009-022

Information Sharing Across Private Databases: Secure Union and Intersection Revisited

by

Pawel Jurczyk, Li Xiong

Mathematics and Computer Science

EMORY UNIVERSITY

Information Sharing Across Private Databases: Secure Union and Intersection Revisited

Pawel Jurczyk
Department of Math and CS
Emory University
Atlanta, Georgia 30322
Email: pjurczy@emory.edu

Li Xiong
Department of Math and CS
Emory University
Atlanta, Georgia 30322
Email: lxiong@emory.edu

***Abstract*—There is a growing demand for sharing information across multiple autonomous and private databases. The problem is usually formulated as a secure multiparty computation problem where a set of parties wish to jointly compute a function of their private inputs such that the parties learn only the result of the function but nothing else. In this paper we analyze existing and potential solutions for secure multiparty computation protocols for union and intersection. We also present an alternative random shares based approach and show that the protocols, although quite simple, are more efficient than existing protocols while providing reasonable level of security that can be adjusted by users. We formally analyze the security properties and the cost of our protocols. We also experimentally compare the performance of our approach with the existing solutions.**

I. Introduction

The amount of personal or sensitive information stored in multiple distributed databases is constantly growing. Institutions increasingly recognize the critical value and opportunities in sharing such a wealth of information. Due to privacy and security constraints, however, the institutions are not willing to disclose their private data to others. The problem is usually formulated as a secure multiparty computation (SMC) or distributed privacy preserving data sharing problem [14], [19] where a set of parties wish to jointly compute a function of their private data inputs such that the parties learn only the result of the function but nothing else.

In this paper, we focus on the union and intersection problem, in which multiple entities wish to collaborate and share the union and intersection of their data without disclosing anything else. For secure union, all the data records will be revealed as part of the result, however, the owner of a certain data record shall not be disclosed. For secure intersection, only the common data records shall be disclosed. We assume the *semi-honest* (or *honest but curious*) adversary model commonly used in secure SMC problems. A semi-honest party follows the rules of the protocol, but it can attempt to learn additional information about other nodes by analyzing the data received during the execution of the protocol. The *semi-honest* model is realistic for our problem scenario where multiple institutions are collaborating with each other to get the correct result for their mutual benefit. Below we describe two scenarios that motivate the secure union and intersection operation under the semi-honest assumption.

Secure Union Scenario: Genomic Information Sharing. Consider a scenario in [20] for genomic data sharing. A patient, John Smith, visits a local hospital and is diagnosed via DNA diagnostic test with some DNA-influenced disease. After the visit, the hospital stores both, clinical and medical data in its local database. Next, John visits few other hospitals for treatment where his medical data and DNA are also collected and stored. For research purposes, the hospitals forward their DNA databases to a research group for sharing. While the sequences are only tagged with pseudonyms of patients, the submitting institution of the DNA records are disclosed. As a result, if an adversary knows which hospitals John Smith visited, called a trail, s/he can track his DNA information by the unique features of the trails. To prevent such a risk, we can use secure union protocol to share the DNA information such that the contributing institutions of the DNA records are not disclosed.

Secure Intersection Scenario: Selective Customer Information Sharing. Consider a few insurance companies that provide home or auto insurance. The companies might be interested in finding an intersection of their customers. This can be driven by various factors, such as identification of fraudulent policy claims or verification of history of clients. Any company cannot send its whole dataset of clients to other companies due to security and competition reasons. In this case, a secure intersection algorithm can be used to identify a set of common clients between the companies and the companies can then share only relevant data about those common customers.

Contributions. In this paper we analyze existing representative secure union and intersection protocols. We also analyze anonymous communication protocols as a potential solution for the secure set operations. In addition, we present an alternative simple yet effective protocol based on random shares approach. In contrast to traditional SMC protocols, the approach we suggest achieve sufficient (but not absolute) security for participating parties at much lower cost for practical usage. We present a set of formal analysis evaluating and comparing the protocols in terms of their complexity, security characteristics and cost. We also implemented all the protocols including the existing ones and experimentally evaluate their cost. Our goal

in this paper is not to promote specific protocols, but to: 1) systematically analyze and experimentally evaluate existing protocols, and 2) demonstrate that simple solutions exist if we make a tradeoff between security, efficiency and accuracy and they may be desirable for certain practical settings.

Organization. The remainder of this paper is organized as follows. Section II briefly reviews related work. Sections III and IV discuss and analyze existing and potential protocols for secure union and intersection. Section V presents the random shares secure union protocol with its analysis followed in Section VI. Section VII introduces secure intersection protocol along with its analysis. Section VIII gives a set of experimental evaluations. Finally, Section IX concludes the paper.

II. Related Work

In the problem of secure multi-party computation (SMC) [11], [14], [19], a given number of participants, each having a private data, wants to compute the value of a public function. A protocol is *secure* if, at the end of computation, all participants know only their local inputs and the final result. It is important to note that for practical purposes we may relax the security goal for a tradeoff for efficiency. Although a general solution to SMC problems has been proven to exist for any function, its high computational overhead makes it impractical. Specialized protocols have been proposed for various functions such as sum [21], the kth element [2], set intersection and intersection size [3], and set union [16], [7].

A closely related and more recent research area is privacy preserving data mining and sharing across distributed data sources [8], [24], [19]. It follows the secure multi-party computation model and the main goal is to ensure that data is not disclosed among participating parties while allowing certain mining or querying task to be carried out. Specialized protocols are designed for various mining tasks with varying degree of accuracy, security, and cost (e.g. [18], [23], [12], [3], [16], [27], [26], [25]).

Most above work assume an honest or semi-honest adversary model [14]. Other works consider broader threat space including malicious adversaries [29], [4], [17], [15].

III. Analysis of Existing and Potential Protocols for Secure Union

Secure union problem definition. Given n ($n \geq 2$) sites, each site holding a local set of data tuples or items x_i, we wish to compute $X = \bigcup x_i$ while minimizing the probability of a node revealing its ownership of x_i to other nodes. Note that secure union does not have practical sense when there are only 2 parties.

Various solutions for computing set union were proposed in the literature. They generally fall into three categories: general circuit-based protocols, specialized cryptography-based protocols, and probabilistic protocols. We briefly describe them or their variants below. We discuss their adaptability from set union to bag union or vice versa. When available, we cite the analysis results from the original papers. Otherwise, we conduct an analysis and present our results.

In addition, anonymous communication protocols, while not directly designed for secure multi-party computations, can be also used for set operations. We describe how to adopt them for secure set operations and analyze their security and cost.

A. Circuit-Based Secure Union

The secure union can be implemented using secure circuit evaluation [1]. Yao [28] showed that any multi-party computation task can be solved by building a combinatorial circuit, and simulating that circuit by participating nodes. The secure union can be computed as follows. First, each node is creating a bit vector with as many bits as there are items in the domain. We will assume that the domain of items is denoted by M. Therefore, each site will have a vector V_i with the length of $|M|$ bits. Next, the nodes generate a circuit that computes bitwise OR operation on all the vectors. The result of such a circuit is a vector that will represent result of the union operation.

The circuit-based algorithm defined above can be modified to compute a bag union as well. The only change that has to be done is to modify the circuit so that instead of calculating the OR, it will calculate sum. The result of such a protocol will be a vector of numbers representing item multiplicity rather than a vector of bits.

The circuit-based protocol, although is provably secure, is computationally prohibitive in practice. First, the size of a circuit depends on the domain size for the data items. For larger domains the circuit calculation can take very long. Second, the size of data being transferred between nodes does not depend on size of the result, but on the domain size. As a result, the cost of secure circuit generation and evaluation add significant overhead.

Cost. We estimated the cost of communication and computation for a semi-honest variant of Yao's protocol using a similar analysis as the one presented in [3]. The number of gates the protocol requires is $n(|M|)G_e$ and the corresponding communication and computation costs are $4k_0n(|M|)G_e$ and $2C_rn(|M|)G_e$, respectively, where k_0 is the size (in bits) of keys used for circuit gates, G_e is the number of gates required to compare 2 numbers, and C_r is the cost of pseudorandom function evaluation.

B. Cryptography-Based Secure Union

As the general circuit-based solution is extremely expensive, specialized cryptography-based protocols are proposed for the union operation based on commutative encryption schemes[8], [16], [7] or homomorphic encryption schemes and polynomial representation of sets [9]. We will focus on protocols based on commutative encryption as a representative for this category of protocols. A given encryption protocol is said to be commutative, if for given encryption keys $K_1, K_2, ..., K_n \in K$ and any permutation i, j the following equations hold:

$$\forall_{m \in M} E_{K_{i_1}}(...E_{K_{i_n}}(m)...) = E_{K_{j_1}}(...E_{K_{j_n}}(m)...) \quad (1)$$

and $\forall_{m_1,m_2 \in M}$ there exists k such that:

$$Pr(E_{K_{i_1}}(...E_{K_{i_n}}(m_1)...) = E_{K_{j_1}}(...E_{K_{j_n}}(m_2)...)) < \frac{1}{2^k} \quad (2)$$

Given the commutative encryption scheme, secure union protocol can be implemented as follows [16]. First, all nodes are arranged in a ring. Each site encrypts its own items using its own encryption key and ships the result to the next site in the ring. Each site then encrypts the received items using its encryption key and sends the result to another site and so on. Assuming there are n nodes, in the n-th step, each node receives his items encrypted by itself and all other nodes. Since the equation 1 holds, any duplicate in original items will also be a duplicate in encrypted set and the decryption of the items can occur using decryption keys $K_1...K_n$ in any order. All the nodes send its fully encrypted data to one of the nodes. Then, the selected node calculates union of all the items it received and decrypts those items using its key and sends the result to the next node in the ring. The next node decrypts items and sends the result to the next node and so on. Once every node decrypts every item, the union is found and can be broadcasted to all the interested nodes.

The algorithm above finds a bag-union without revealing which item was contributed by which node. To calculate the set union, one can remove the duplicates in the fully-encrypted set before the decryption phase. This of course would prevent revealing which items are duplicates (node 1 only knows the encrypted format of the duplicate items), however, the number of items that exist commonly between sites would be known.

Security. As proved in [16], the discussed protocol securely computes union, *revealing* a bounded set of innocuous information such as size of the intersection of the data items and number of items at the sites.

Cost. Using a similar analysis as the one presented in [3], we conducted a cost analysis for the protocol. The estimate for the communication cost is $n^2dk(2n+1)$ and the computation cost is $2n^2C_ed$, where n is a number of nodes, d is an average number of items provided by each node, k is the size of encrypted item (in bits) and C_e is the cost of encryption/decryption of an item.

C. Probabilistic Secure Union

A probabilistic secure union algorithm was proposed in [5] to address the concerns of high overhead associated with traditional SMC protocols. The protocol also uses a bit vector V_i to represent the data items at each node and calculates the logical OR of the bit vectors. The main idea is to use r rounds and to use randomization in each round when generating the result of the algorithm. In the protocol, a global vector V is passed from one node to another along the ring (initially V is filled in with random values). When the vector is received from a predecessor, the node performs probabilistic bit flipping in the received vector, and passes the result of this operation to his successor. After r rounds, the vector V contains result of the algorithm and can be broadcasted to all interested nodes.

The algorithm finds a set union and its modification to calculate a bag union can be problematic. In case of a bag union, the intermediate vector V should store counts of items, and thus the probabilistic bit flipping approach is not easily applicable.

Correctness. As shown in [5], the protocol is not deterministic and the result is correct only with certain probability guarantee. For a given number of rounds, $r \geq max(3, -log[1 - \{\frac{8}{7}(1-\epsilon)\}^{1/(n-1)}])$, the probability of having an error in each bit of the result vector is at most ϵ.

Security. The protocol is not absolutely secure and does reveal information about the local data. [5] proved that the probability of one node deducting that its successor has a given data item (a term in the context of the paper) is 0.71. Unfortunately, when nodes collude, this probability is much higher (however, no details are given in the paper).

Cost. We also conducted a cost analysis of the protocol. The estimate for the communication cost is $rn|M|$ and the computation cost is $rn|M|C_c$, where C_c is the cost of evaluating if statements in the protocol.

D. Anonymous Communication-Based Secure Union

Anonymous communication [10] is a technique of bouncing communications around a distributed network of relays in order to prevent, for instance, somebody watching Internet connection from learning what sites one visits or prevent the sites one visits from learning one's physical location. Instead of taking a direct route from source to destination, data packets take a random pathway through several relays. The pathway between source and destination, called circuit, is usually used for some short time and after that a new random circuit is created in order to increase anonymity.

Due to the nature of set union operation and its main goal to protect the anonymity of the data owners, anonymous communication techniques are particularly suitable for implementing secure union computations. The algorithm could simply use an anonymous communication protocol to ship all the data items to a single node. Once this step is completed, the union is found and can be sent to all the interested parties. As the recipient does not know the message originator, the ownership of items in the union is protected. The protocol finds a bag union and it can be also modified to remove duplicates, similarly as the cryptography-based approach.

Security. The protocol described above guarantees security provided that no nodes collude, revealing the size of intersection between nodes (the intersection can be calculated using encrypted items sent to the node computing union) and size of subsets owned by other nodes (protecting the identity of those nodes). If the recipient of data items colludes with some nodes from the communication circuit, the risk of corrupting privacy increases. Such a risk can be greatly minimized by using longer circuits. Note also that the security of the protocol can be further increased. In the description above, even though the exact node is unknown, the recipient gains knowledge about a given set of items being owned by some node. To minimize

this exposure, the nodes can ship data in a few random packets. In the case of set union which removes duplicates, the recipient node learns exact duplicate items (not only the encrypted values).

Cost. The estimate for the communication cost of the protocol is $ndk(c+1)$ and the computation cost is $2nC_edc$, where n, d, k and C_e have the same meaning as in the previous subsection and c is the number of nodes in a circuit.

IV. Analysis of Existing and Potential Secure Intersection Protocols

Secure intersection problem definition. Given n sites ($n \geq 2$), each site holding a local set of data tuples or items x_i, we wish to compute an intersection $X = \bigcap x_i$ while minimizing the probability of a node revealing $x_i \backslash X$ to other nodes.

Similar to secure union, a secure intersection protocol can also be implemented in various ways. Below we describe or adapt various solutions for secure intersection implementation and briefly analyze their security and cost.

A. Circuit-Based Secure Intersection

A circuit-based secure intersection computation was discussed and analyzed in [3]. It uses a slightly different approach than the secure union. The inputs of the circuit have sizes of local datasets x_i, and the output is a bit vector of size of one of the inputs (say, x_j). Note that such an approach is possible due to the different property of intersection operation: an item will appear in the output result if it is present in all the inputs. The circuit will compute bitwise AND operation on all the input vectors.

Cost. The circuit-based secure intersection suffers from significant computation overhead as the circuit-based secure union calculation. However, the size of a circuit for secure intersection does not depend on the number of items in the domain of items. The cost of communication and computation of the algorithm, according to the analysis presented in [3], is estimated as: $4k_0 \prod(|x_i|)G_e$ and $2C_r \prod(|x_i|)G_e$, respectively, where k_0, G_e and C_r have the same meaning as above.

B. Cryptography-Based Secure Intersection

The cryptography-based approach can be used to securely calculate an intersection. An algorithm for 2-party case was proposed in [3]. For a multi-party case, similar to the union algorithm, secure intersection can use the commutative cryptography approach. The approach we discuss here is a variation of the algorithm from [3]. The first phase of the intersection protocol is exactly the same as in the secure union (encrypting all the items by all the nodes). In the second phase, the items are shipped to a single node as well, however, the selected node calculates an intersection between the encrypted sets (as we are using a commutative encryption, duplicates will also be duplicates in the encrypted dataset). Let us assume that in the second phase all the encrypted items are being shipped to node 1. Initially node 1 assumes that the protocol result are node's own items: $r = s_1$. Next, the node updates the initial guess as the items from other nodes are received: $r = r \cap_B s_i$, where s_i are encrypted items received from node i. Once node 1 receives items from all other nodes, the encrypted intersection r is found. To find the actual intersection, decryption has to be performed by all the nodes, similarly as in the secure union calculation. The algorithm can calculate bag intersection or set intersection by performing the corresponding intersection operation in the second phase of the protocol.

Security. The algorithm securely calculates intersection revealing only count of records owned by each node.

Cost. The communication and computation costs are slightly less than the cost of cryptography-based secure union (due to smaller result size), and can be estimated as $ndk(n+2)$ and $nC_ed(n+1)$, respectively.

C. Probabilistic Secure Intersection

A similar idea to the one discussed in section III-C can be used to implement a probabilistic secure intersection protocol. We modify the algorithm for set union and present the algorithm for set intersection in listing 1.

Algorithm 1 Calculation of V at node i in round r

```
1: INPUT: r - round #, v_i - local bit vector, V - intermediate result
2: P_ex ← 1/2^r
3: P_in ← 1 − P_ex
4: for j = 1 to length(V) do
5:     if v_i[j] = 1 and V[j] = 1 then
6:         Set V[j] = 1 with probability P_in
7:     end if
8:     if v_i[j] = 0 and V[j] = 1 then
9:         Set V[j] = 0 with probability P_in
10:    end if
11:    if v_i[j] = 1 and V[j] = 0 then
12:        Set V[j] = 1 with probability P_ex
13:    end if
14: end for
15: Send V to successor(i)
```

The protocol has the same limitations as the probabilistic secure union protocol. The main drawbacks include dependence on the size of the domain (instead of dependence on size of results) and a non-deterministic behavior.

Correctness. The protocol is not deterministic. Similar to [5], we can derive that the minimal number of rounds that guarantees an error in each bit of the result vector to be at most ϵ is $r = log_2\epsilon$.

Security. The protocol also reveals information. Similar to [5], we can derive that the probability of one node deducting that its successor does not have a given item is 0.71.

Cost. We estimated the cost for the above protocol. The communication cost is $rn|M|$ and the computation cost is $rn|M|C_c$, where C_c is a cost of evaluation of if statements in lines 5-7, 8-10 and 11-13.

D. Anonymous Communication-Based Secure Intersection

The secure intersection can be also implemented using an anonymous communication protocol similar to the secure union. An anonymous communication protocol can be used to ship all the data items to a single node. Once this step is

completed, the intersection can be found (either set or bag intersection) and the result can be sent to all parties. Note that the protocol will work provided that each node will send all his data items in a single message. Such a requirement guarantees that the receiving node will be able to identify sets owned by other nodes in a proper way.

Security. The anonymous communication-based secure intersection, unlike secure union, guarantees less data security. The reason lies in the differences between union and intersection operations. For union, each of the items owned by a given node will appear in the final result. Therefore, the receiving node, provided that he does not know the sender, will not see any item that does not appear in the final result set and he does not learn anything new. For intersection, however, not all the items owned by a given node appear in the final result set. Therefore, sending all the node's items to another node reveals the knowledge about items that will not appear in the final result.

Cost. The communication and computation costs of the protocol are the same as the costs for anonymous communication-based secure union.

V. Random Shares Based Union Protocol

In this section we present our set union protocol that uses a random shares approach similar to that of a secure sum protocol. The protocol we describe computes a bag union. However, the protocol can also be modified to remove duplicates if necessary.

To facilitate the discussion, we first describe a simple secure sum protocol to illustrate the random shares idea and then present the details of our random shares secure union protocol.

Secure sum. The secure sum protocol works as follows [8]. Assume that the sum value is known to lie in the range $[0..m]$ and that all the participating nodes are arranged in a ring. The first node generates a random number r and passes to the second node value $v_1 = (x_1 + r)\ mod\ m$. The second node receives the value v_1 from the first node, computes $v_2 = (v_1 + x_2)\ mod\ m$ and passes v_2 to the third node and so on. The last node sends value v_n to the first node. Then, the first node can subtract r from the value v_n it receives and the sum is known. The protocol can be modified in order to address the problem of colluding nodes. Instead of using only one round, it can use p rounds. In each round the ring is permuted and, instead of adding its x_i to the intermediate result, each node adds a random share of x_i. The random shares have to satisfy the following: $\sum_{j=1}^{p} s_{i_j} = x_i$ (s_{i_j} denotes a share contributed by node i in round j).

Random shares based secure union. Now we present the *secure union protocol* utilizing a similar random shares based approach. Our main design goal for the protocol is to be able to make a tradeoff between security and efficiency so that it can achieve reasonable and probabilistically bounded security at a much lower cost. Note that the probabilistic secure union protocol discussed in III-C also gives a probabilistic security bound and allows tradeoff between security and efficiency as well as accuracy. In contrast, our protocol is designed to be deterministic in terms of accuracy and its cost does not depend on the domain size.

There are three key ideas to the protocol. First, each node introduces random items so that it will not suffer from a provable exposure of its ownership of items. Second, a starting node is randomly selected so that nodes close to the starting node on the ring will not suffer from a high probability of data disclosure. Finally, the protocol uses multiple rounds and for each round the nodes are permuted and each node participates with a random share of its data items. This random shares based approach further minimizes the effect of potential collusion of the nodes.

Algorithm 2 Random shares secure bag union protocol.

```
1: INPUT: x_i: local subset contributed to union
2: Generate random set r_i, choose leader, set IR ← ∅
3: Phase1
4: for k = 1 to p do
5:     Arrange nodes in a ring topology randomly
6:     if leader then
7:         Send IR ∪_B x_{i_k} ∪_B r_{i_k} to successor
8:         Receive IR from predecessor
9:     else
10:        Receive IR from predecessor
11:        Send IR ∪_B x_{i_k} ∪_B r_{i_k} to successor
12:    end if
13: end for
14: Phase 2
15: Arrange nodes in a ring topology randomly
16: if leader then
17:    Send IR −_B r_i to successor
18:    Receive IR from predecessor
19:    Result ← IR
20: else
21:    Receive IR from predecessor
22:    Send IR −_B r_i to successor
23: end if
```

The protocol works as follows. First each node i generates a random set r_i and leading site l is chosen randomly. Next the p rounds of the protocol begin where each node adds its random share to the intermediate result. In each round, all the nodes are arranged in a ring topology randomly. This can be done for instance by selecting a random number t_i by each node. Then, the nodes can be arranged in the order indicated by growing values if t_i using a secure k-th element algorithm [2]. Once the nodes are arranged, the leading site adds a random share of its local set x_l and a random share of its random set r_l to the intermediate result from the previous round and passes the result to its successor. The other nodes performs the computation similarly. When node l receives the result from its predecessor, the next round begins. Note that each node has to choose random shares so that: $\cup_{j=1}^{p} x_{i_j} = x_i$ and $\cup_{j=1}^{p} r_{i_j} = r_i$ where x_{i_j} and r_{i_j} denote a random share of the data items and random items added to the intermediate result by node i in round j. When p rounds are completed, the protocol moves to the next phase.

In the second phase the new random ring topology is generated (note that the leading site remains the same though). Next, the leading site l subtracts its random items r_l from

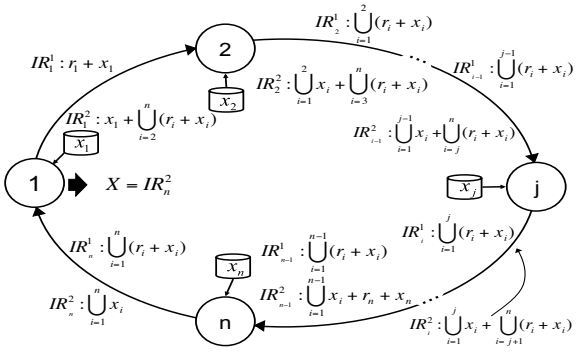

Fig. 1. Random shares set union protocol (single-round version). IR^1/IR^2 represent intermediate result in the first/second phase.

intermediate results received in the previous phase and passes the result to the successor. Then, each node *i* subtracts its local random set r_i from the intermediate result and passes the result along the ring. When node *l* receives the result from its predecessor, the protocol finishes and the union is found. To further enhance the security of the protocol, one could also use p rounds for the second phase. A sketch of the algorithm is presented in Algorithm 2 and Figure 1 presents phases of the protocol when only one round is used ($p = 1$).

An important issue in the protocol is the random data item generation. The questions we need to answer are: 1) how to generate a good random set r that look legitimate to other nodes and are indistinguishable from real data, and 2) what should be the size of r? We defer the second question to the next subsection when we analyze the protocol in detail and briefly discuss the first question here. There are a number of factors that need to be considered for generating legitimate items. First, the random item has to come from a legitimate domain. For numeric attributes, we assume the domain range is known to all the nodes. For discrete attributes with closed set of values (such as geographic entities), well-known dictionaries can be exploited. Second, we have to follow the underlying distribution of the data. Most attributes, such as age or weight, can be expected to follow normal distribution and thus a random item can be generated using such distributions. Finally, a good random item generation also needs to take into account correlation between attributes (one can expect that the weight and height values are strongly correlated). Therefore, a node can perform certain correlation analysis on its local data and generate attribute values based on their dependencies.

VI. Analysis of Random Shares Secure Union

In this section we analyze our protocol in detail. We first introduce the security metric that we use for evaluating how well we achieve our security goal and present a formal analysis using this metric. We will plot the analytical bounds derived in this section along with our experimental results in the experiment section.

A. Security Metric and Attack Models

Our security goal is to prevent an adversary from being able to determine the ownership of items from the final result. Given the goal, we need to quantify the degree of data exposure for each node. We adopt the loss of privacy (*LoP*) metric [26] for this purpose. Let *R* denote the final result of the algorithm and *IR* denote the intermediate result during execution of the protocol. Suppose an adversary, as an attack, makes a claim C about the data at a node, we define two probabilities. The first, $P(C|IR, R)$, is the probability of claim *C* being true when node has both *IR* and *R*. The second, $P(C|R)$, is the probability of claim *C* being true when node has only the final result *R*. The loss of privacy is defined as follows:

$$LoP = P(C|IR, R) - P(C|R) \tag{3}$$

It essentially measures the difference between the posterior probability (with intermediate result) and the prior probability (without intermediate result) with respect to an attack. This, in spirit, is similar to the metric presented in [13], [22] which measures the information disclose for anonymization using the notion of posterior probability (with published dataset) and the prior probability (with published dataset).

In our case, there are two kinds of data exposure (or attacks) with corresponding claims that can be made by an adversary, namely, *set exposure* and *item exposure*. For set exposure, an adversary is able to make a claim on the whole set of items a node contributes to the final union result (C = *node i contributed subset a_i to the final result*). For item exposure, an adversary is able to make a claim on a particular item a node contributes to the final result (e.g. C = *node i contributed item v_i to the final result*). Below we analyze the loss of privacy for these two attacks respectively.

B. Security Analysis

We focus our analysis on single-round versions of the protocol ($p = 1$). Increasing the number of rounds will only increase the security of our solution.

Theorem 1. The Loss of Privacy (LoP) for the random shares based union protocol with respect to set exposure attack is bounded by:

$$LoP = \frac{1}{n-1} * \left(\frac{m - c + c_1}{m}\right)^{|r|} \tag{4}$$

If k sites collude, the LoP is bounded by:

$$LoP = max(\frac{2(k-1)(n-k)}{(n-1)(n-2)} * \left(\frac{m - c + c_1}{m}\right)^{|r|}, \frac{2k(k-1)^2}{n^3}) \tag{5}$$

Proof. As the worst case scenario, we consider the starting node (we assume node 1 is the starting node) as the victim and the second node (node 2) as the adversary. To begin with, we also assume that node 2 is aware that node 1 is the starting node. This is the worst case because node 2 only has to identify a set of real data items (not randomly generated items) from the intermediate result it receives from node 1 while any

further adversary nodes will have to not only identify the real items, but also the owner of items.

Suppose node 2 is trying to identify the whole set of items of node 1 by making a claim C: $x_1 = a_1$. We estimate $P(C|R)$ and $P(C|IR, R)$ below and derive the LoP.

We start by computing $P(C|R)$, the probability of the claim being true given only the final result X. Given only X, the best an adversary (node 2) can do for guessing x_1 is to select a random number of items from X which are not among his own items x_2. If we assume that X contains c distinct items, the probability the claim is true is given by (note that x_1 can contain any number of items, so the adversary has to guess the size of this set and the items):

$$P(C|X) = \frac{1}{\sum_{a=0}^{c} \binom{|X - x_2|}{a}} \approx 0 \quad (6)$$

We are limiting the analysis above to the case when result set contains only distinct items. As having duplicates helps an adversary, we are actually finding a lower bound for the probability $P(C|R)$ (and an upper bound for the *LoP*).

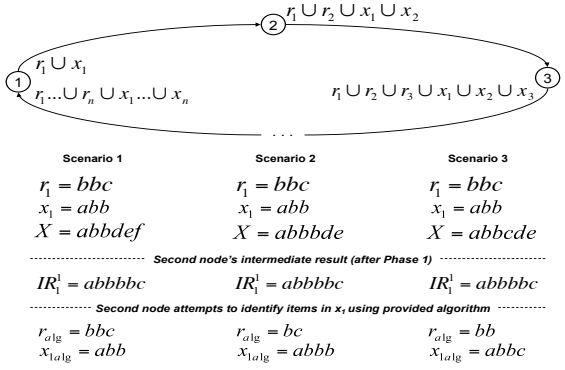

Fig. 2. Example of data exposure of node 1 to node 2 (r_{alg} - guessed r, x_{1alg} - guessed x_1, IR_1^1 - intermediate result in Phase 1)

We now compute $P(C|IR, R)$, the probability of the claim being true given the intermediate result IR_1^1 and the final result X. The intermediate result IR_1^1 contains the random set r_1 generated by node 1 and the subset x_1 contributed by node 1. If it happens that no item from the random set r_1 appears in the final result set, node 2 can determine r_1 using $r_1 = IR_1^1 - X$ and consequently determine x_1 using $x_1 = IR_1^1 - r_1$. Thus the best an adversary can do for guessing x_1 is to make a claim C: $x_1 = IR_1^1 - (IR_1^1 - X)$. Figure 2 presents a few possible scenarios for the claim. The probability of this claim being true can be derived as follows:

$$\begin{aligned} P(C|X, IR_1^1) &= P(x_1 = r_1 \cup x_1 - (r_1 \cup x_1 - X)) \\ &= P(x_1 = r_1 \cup x_1 - (r_1 - (X - x_1))) \quad (7) \\ &= P(r_1 \cap (X - x_1) = \emptyset) \end{aligned}$$

If we assume that the domain M of items contains m distinct items and if we again assume that the result of union algorithm contains c distinct items and that node 1 contributes c_1 distinct items to the final set, the probability of a random set r_1 not containing any item from the set $X - x_1$ is given by:

$$P(r_1 \cap (X - x_1) = \emptyset) = \left(\frac{m - c + c_1}{m}\right)^{|r_1|} \quad (8)$$

Now we can derive the LoP for the starting node (given the knowledge of the starting node by the adversary) as:

$$LoP = \left(\frac{m - c + c_1}{m}\right)^{|r|} - 0 = \left(\frac{m - c + c_1}{m}\right)^{|r|} \quad (9)$$

The above analysis assumed that node 2, the adversary node, is aware that node 1 is the starting node. As our protocol utilizes a randomized starting scheme, the probability of a node being the starting node is $\frac{1}{n-1}$ (assuming an adversary node is not the starting node). Thus we derive the bound of LoP as presented in equation 4.

Now let's turn our attention to the case of two nodes colluding in order to identify items provided by some other node. We will assume that in the ring nodes $i-1$ and $i+1$ collude in order to identify items provided by node i. To alleviate the problem of nodes collusion, the fact that each node generates random items helps to limit the LoP. If the nodes collude in the first phase of the protocol, they can identify set $(x_i + r_i)$ using the following formula: $(x_i + r_i) = IR_{i+1}^1 - IR_{i-1}^1$. If in the second phase nodes do not end up in the same positions in the ring, the best they can do is to proceed according to the algorithm. As the ring is generated randomly, the probability of two colluding nodes being arranged in the way that makes this attack possible is $\frac{2}{n-1}$ (for the attack to be possible the first colluding node can be placed in any place in the ring; then the second colluding node can be placed two positions before the first one or two positions after). In this case, the analysis will be similar to the analysis above with respect to the attack from the second node guessing the set provided by the first node, and LoP can be estimated using equation 9 as:

$$LoP = \frac{2}{n-1} * \left(\frac{m - c + c_1}{m}\right)^{|r|} \quad (10)$$

If colluding nodes surround the same node in the first and second rounds, the privacy of the surrounded node can be clearly compromised, as the whole set provided by this node can be identified. The probability of such scenario can be estimated as follows: $\frac{2}{n-1} * \frac{2}{n-1} * \frac{1}{n-2}$ (the colluding nodes have to surround the same node in the first and second phases). For larger n the probability can be approximated as $\frac{4}{n^3}$ and the LoP can be estimated as:

$$LoP = max(\frac{2}{n-1} * \left(\frac{m - c + c_1}{m}\right)^{|r|}, \frac{4}{n^3}) \quad (11)$$

When more than two sites collude, the probability of one of the scenarios analyzed above is higher. Let's assume a general case of k colluding sites. The first scenario we analyzed above was when colluding nodes surrounded the attacked node only in the first phase. This probability can be calculated as $\frac{2(k-1)(n-k)}{(n-1)(n-2)}$ (any 2 of the k nodes can

surround a node that will be attacked). On the other hand, the probability of surrounding attacked node by colluding sites in both phases of the union algorithm can be estimated as follows: $\frac{2(k-1)(n-k)}{(n-1)(n-2)} * \frac{k}{n-1} * \frac{k-1}{n-2}$ which for larger n can be estimated as $\frac{2k(k-1)^2}{n^3}$. Thus, the LoP when k sites collude is bounded by eq. 5.

Theorem 2. The LoP of the protocol with respect to item exposure is 0 if no sites collude. If k sites collude, the LoP is:

$$LoP = max(\frac{2(k-1)(n-k)}{(n-1)(n-2)} * \frac{|x_i|+|x_i \cap (r \cap (x-x_c))|}{|x_c|+|(r \cap (x-x_c))|}, \frac{1}{n-1}, \frac{2k(k-1)^2}{n^3}) - \frac{1}{n-1} \quad (12)$$

Where i is an attacked node and x_c is a union of items contributed by colluding sites.

Proof. We again consider the starting node (node 1) as the victim and the second node (node 2) as an adversary. Suppose node 2 is trying to identify a single item contributed by node 1, x_1, by making a claim C: $v_1 \in x_1$.

We first compute $P(C|R)$. Given only X, the best an adversary can do for guessing a single item in x_1 is to select an item from X. The probability this claim being true is: $P(C|R) = \frac{1}{n-1}$.

We now compute $P(C|IR, R)$, the probability of the claim being true given the intermediate result IR_1^1 and the final result X, and analyze how the intermediate result can help this node to find the owner of some items. Using a similar approach as in the set exposure scenario, the adversary can try to find items contributed by its predecessor. Simply put, the less items from random set r_1 are in the algorithm's final result, the easier the second node can identify items contributed by the first node. Specifically, observing again the scenarios presented in Figure 2, and assuming that node 2 is aware of node 1 being the leader, the probability $P(C|IR, R)$ is given by (we assume that each node contributes in average $\frac{c}{n}$ items):

$$P(C|IR,R) = \frac{|x_1| + |x_1 \cap (r_1 \cap (X - x_2))|}{|x_2| + |(r_1 \cap (X - x_2))|} \leq \frac{2 * \frac{c}{n}}{\frac{c}{n} + |r_1| * \frac{c-\frac{c}{n}}{m}} \quad (13)$$

The worse case LoP for the starting node can be then derived as follows:

$$LoP \leq \frac{2 * \frac{c}{n}}{\frac{c}{n} + |r| * \frac{c-\frac{c}{n}}{m}} - \frac{1}{n-1} \quad (14)$$

Considering the randomized starting scheme, the expected LoP for the item exposure is as follows:

$$LoP = max(\frac{1}{n-1} * \frac{2*\frac{c}{n}}{\frac{c}{n}+|r|*\frac{c-\frac{c}{n}}{m}}, \frac{1}{n-1}) - \frac{1}{n-1} = 0 \quad (15)$$

The analysis of a scenario with collusion between nodes is quite similar to that for the set exposure. If colluding nodes surround a victim node only in the first phase, the fact of generating random items by each node brings the analysis to a similar scenario as discussed above. On the other hand, if colluding nodes surround the same node in both rounds, all the items of the attacked node can be compromised. If there are k colluding nodes, the overall LoP can be estimated as in equation 12.

Comparison with probabilistic secure union. It is worth comparing our analysis with the probabilistic union protocol since both of them give a probabilistic security bound. The probability bound derived in [5] for the probabilistic union protocol, in fact, corresponds to our definition of $P(C|IR, R)$ with respect to item exposure when there is no collusion. So the LoP for the item exposure of probabilistic union protocol without collusion can be estimated as $0.71 - \frac{1}{n-1}$.

Generating random items. One remaining issue is how many random items a node should generate in the first phase. Now we will attempt to devise rules which can guarantee certain LoP bounds. Our analysis is again divided into two cases, namely, set exposure and item exposure.

Set exposure. Given Equation 4 that gives the *LoP* bound for set exposure, we can obtain the following:

$$|r| = \lceil \log_{\frac{m-c+c_1}{m}} (n-1) * LoP_{expected} \rceil \quad (16)$$

Equation 16 allows one to find a minimal number of random items that should be generated in order to guarantee a given LoP bound with respect to set exposure when no collusion is assumed. The minimal number of random items in the case of collusion can be similarly estimated using the factor of equation 11 that depends on size of r_i. In this case, however, the minimal LoP cannot be smaller than the factor $\frac{k(k-1)}{n^3}$ which does not depend on the size of the random set.

Item exposure. The *LoP* bound with respect to item exposure was presented in equation 15. This equation shows that in average the algorithm does not reveal any additional information. Despite this fact, however, one has to deal with worst case *LoP* presented in equation 14 (the case when the first node in the ring chooses the highest value of random value t). In this case, from Equation 14 we can derive the following:

$$|r| = \lceil \frac{m * \frac{c}{n}}{c - \frac{c}{n}} * \left(\frac{2}{LoP_{expected} + \frac{1}{n-1}} - 1 \right) \rceil \quad (17)$$

Equation 17 allows one to find a minimal number of random items that should be generated in order to guarantee a given LoP bound for item exposure when nodes do not collude. To estimate a minimal number of random items that need to be generated in case of collusion, equation 12 can be used.

C. Cost Analysis

The communication and computation costs of the protocol are $kn(\frac{d}{2} + \frac{3}{2}nd + nr)$ and $nsC_s(2p+1)$, respectively, where k is an average size of item, d is an average number of items owned by node, r is size of random set generated by each node and C_s is a cost of set operation (union/intersection/difference) on two input sets.

VII. Random Shares Based Intersection Protocol

Using the similar random shares idea and the same assumptions as in section V, we can also implement a general secure intersection protocol. However, there are some subtle differences between secure union and secure intersection that require some modifications to the protocol. In the secure union, any node i is certain that the items he owns will appear in the final result. Secure intersection is different in that the node does not have such knowledge, as the intersection can contain any subset of the items a node owns.

The random shares based intersection protocol can be implemented, however, using the functionality provided by the random shares secure union as a building block. The protocol can work as follows. In the first phase, each node generates its set of random items r_i. Next, the leader node is chosen. The leader node i starts by calculating $x_i \cup_B r_i$ and passing the result to its successor. The other nodes perform the computation similarly. Once the leader node receives the intermediate result from the last node in the ring, the following is known: $\bigcup_{i=1}^{n}(x_i \cup_B r_i)$. The leader continues broadcasting this set to all the nodes that participate in the protocol.

At this point the nodes begin the second phase of the algorithm which has a goal to remove items that do not belong to the intersection. For this purpose, we use the property of the intersection operation: if a given item does not appear in at least one subset, it will not appear in the result. Using this property, the bag intersection algorithm will remove all the items that will not appear in the final result. This can possibly take few rounds and in each round the subset that can be removed from IR is computed using the random shares based set union protocol. The random shares secure bag intersection algorithm is presented in listing 3. Note that the removal phase finishes when the result of secure set union is empty, which means that the intersection is found.

Algorithm 3 Random shares secure bag intersection protocol.

```
1: INPUT: x_i: local subset contributed to intersection
2: Generate random set r_i, choose leader, set IR ← ∅
3: if leader then
4:     Send IR ∪_B x_i ∪_B r_i to successor
5:     Receive IR from predecessor
6:     Broadcast IR
7:     S ← ∅
8:     repeat
9:         S ← secureSetUnion(IR −_B x_i)
10:        Broadcast S
11:        IR ← IR −_B S
12:    until S = ∅
13:    Result ← IR
14: else
15:    Receive IR from predecessor
16:    Send IR ∪_B x_i ∪_B r_i to successor
17:    IR ← receivedFromLeader
18:    repeat
19:        S ← receivedFromLeader
20:        IR ← IR −_B S
21:    until S = ∅
22: end if
```

It is worth mentioning that finding a set intersection is a special case of the bag intersection algorithm above. In order to compute a set intersection, a bag union and bag difference operations in lines 4, 11, 16 and 20 should be replaced with set union and set difference operations. Moreover, the set intersection will always use the secure union protocol only once.

A. Correctness Analysis

Theorem 3. The above protocol computes set intersection and bag intersection correctly.

Proof. First we show that set intersection works correctly. Our set intersection algorithm first computes $\bigcup_{i=1}^{n}(x_i \cup r_i)$ and then subtracts the union $\bigcup_{i=1}^{n}(\bigcup_{j=1}^{n}(x_j \cup r_j) - x_i)$. Below we show that the calculation effectively leads to a calculation of the intersection of subsets x_i:

$$
\begin{aligned}
\bigcup_{i=1}^{n}(x_i \cup r_i) - \bigcup_{i=1}^{n}(\bigcup_{j=1}^{n}(x_j \cup r_j) - x_i) &= \\
\bigcup_{i=1}^{n}(x_i \cup r_i) - \bigcup_{i=1}^{n}((\bigcup_{j=1}^{n} x_j \cup \bigcup_{j=1}^{n} r_j) - x_i) &= \\
\bigcup_{i=1}^{n}(x_i \cup r_i) - \bigcup_{i=1}^{n}((\bigcup_{j=1}^{n} r_j - x_i) \cup (\bigcup_{j=1}^{n} x_j - x_i)) &= \\
\bigcup_{i=1}^{n}(x_i \cup r_i) - (\bigcup_{i=1}^{n}(\bigcup_{j=1}^{n} r_j - x_i) \cup \bigcup_{i=1}^{n}(\bigcup_{j=1}^{n} x_j - x_i)) &= \\
\bigcup_{i=1}^{n}(x_i \cup r_i) - (\bigcup_{j=1}^{n} r_j - \bigcap_{i=1}^{n} x_i) - (\bigcup_{j=1}^{n} x_j - \bigcap_{i=1}^{n} x_i) &= \\
(\bigcup_{i=1}^{n}(x_i \cup r_i) - \bigcup_{j=1}^{n} r_j) \cup \bigcap_{i=1}^{n} x_i - (\bigcup_{j=1}^{n} x_j - \bigcap_{i=1}^{n} x_i) &= \\
((\bigcup_{i=1}^{n}(x_i \cup r_i) - \bigcup_{j=1}^{n} r_j) \cup \bigcap_{i=1}^{n} x_i - \bigcup_{j=1}^{n} x_j) \cup & \\
(\bigcap_{i=1}^{n} x_i \cap ((\bigcup_{i=1}^{n}(x_i \cup r_i) - \bigcup_{j=1}^{n} r_j) \cup \bigcap_{i=1}^{n} x_i) &= \\
\bigcap_{i=1}^{n} x_i &
\end{aligned}
$$

The proof for correctness of a bag intersection algorithm follows the above transformations. In the bag intersection, all the calculations, except for the call to a secure set union, use bag operations. This guarantees that the multiplicity of items will be taken into account. The fact that secure set union is used to calculate the set S in Algorithm 3 guarantees that the set difference calculation (lines 11 and 20) does not remove more than necessary from the IR. The protocol will finish once the union of sets $IR -_B x_i$ is empty (i.e., the intersection was found).

B. Security Analysis

Theorem 4. The LoP of the random shares based secure intersection protocol with respect to the set exposure attack is the same as LoP of random shares based union protocol.

The LoP with respect to the item exposure attack is defined as:

$$LoP_{intersect} = max(\frac{|x_l|}{|x_l \cup r_l|} * \frac{1}{n-1}, LoP_{union}) \quad (18)$$

with LoP_{union} being the item exposure LoP bound introduced by the random shares union protocol and l being the leader node.

Proof. The analysis will be divided into two parts. First, we will focus on the first phase of the algorithm where the nodes calculate union of subsets $x_i \cup r_i$. Then we will analyze the security of the phase that removes the items from the set IR to identify the intersection. Similar to the analysis of the set union, we will consider the first node as victim as the data exposure at this point will be the highest. When the second node receives the intermediate result from the first node, it receives a set $x_1 \cup r_1$. Assuming that the set contains c items, the probability of identifying the whole set contributed by the first node by the second node therefore can be estimated as follows (assuming r_1 to be sufficiently large):

$$P(C|X) = \frac{1}{\sum_{a=0}^{c} \binom{|x_1 \cup r_1|}{a}} \approx 0 \quad (19)$$

The above equation defines a probability of set exposure. Note that the final result, X, will not be particularly useful for an adversary in a secure intersection protocol as it may not contain all the items from x_1.

The second phase of the algorithm calls the secure union protocol, and the LoP metric of the secure union protocol was analyzed in Section VI. As a consequence, LoP with respect to set exposure introduced in this phase is the same as the set exposure for the union protocol (equation 4). Therefore, the overall set exposure LoP of the secure intersection protocol can be estimated as follows:

$$LoP_{intersect} = max(0, LoP_{union}) = LoP_{union} \quad (20)$$

where LoP_{union} is the set exposure LoP introduced by the union protocol.

We will now consider item exposure attacks. In the first phase the second node can guess that any of the items in the intermediate result received from the first node belongs to that node. The guess will be true with the probability $\frac{|x_1|}{|x_1 \cup r_1|}$. Therefore, the item exposure is bounded by the number of generated random items, and the randomized start of the algorithm reduces this probability by the factor $\frac{1}{n-1}$. The collusion between nodes can help adversaries to some extent in the first phase of the algorithm. If colluding nodes surround node i, they can easily identify the set $|x_i \cup r_i|$. In this case, the LoP analysis is the same as the analysis above. The item exposure of the second phase is again the same as item exposure of the union protocol (equation 15). Thus, the overall item exposure LoP of the protocol can be estimated as presented in eq. 18.

Parameter name	Description	Default value
m	Size of domain	100,000
n	Number of participating nodes	20
c	Size of algorithm result	1000
r	Number of generated random items	varies

TABLE I
SIMULATION PARAMETERS.

C. Cost Analysis

The communication and computation costs of the protocol can be estimated as $\frac{n+1}{n}n(d+r) + s(C_U + n(d+r))$ and $2nC_s + sC_{U_{CPU}}$, respectively, where k, d, r and C_s have the same meaning as in union protocol cost analysis, s is a number of calls to secure set union, and C_U and $C_{U_{CPU}}$ are the communication and computation costs of the set union.

VIII. Experimental Evaluation

In this section we will present a set of experimental evaluations of the proposed protocols. The questions we attempt to answer are: 1) How does the secure set union protocol perform in various settings and how does it compare with the analytical results, and 2) What is the cost of our protocols in comparison to other options?

A. Security of Random Shares Union Protocol

We have implemented the random shares based protocols for both secure union and secure intersection. To answer the first question above, we prepared a simulation of a distributed environment and used synthetically generated data with varying parameters which allowed us to test and evaluate the protocol in multiple scenarios and settings. A summary of the set of simulation parameters is presented in Table I. In all the experiments the default values are used unless otherwise specified. We have assumed that nodes contribute in average $\frac{c}{n}$ items to the final result. We report the results for both set exposure and item exposure attacks. Our experiments assume no collusion scenario. We believe such a setting is sufficient to demonstrate the correctness of our analysis. Moreover, the collusion of a small number of nodes will not increase LoP by significant factor.

Set exposure. The first part of the experimental results is focused on the set exposure attacks. Figure 3 presents the analytical LoP bound (recall Equation 9) and the actual LoP obtained from the experiments when a given number of random items is generated. As can be noticed, given the default number of nodes being 20, even when no random items are generated, the algorithm provides quite high security (expected LoP is 0.05) by utilizing the inherent anonymity of the network of nodes. Generation of only 100 random items by each node (which is 0.1% of the size of domain) causes actual LoP to drop below the level of 0.02. Given a smaller number of nodes, generation of random items becomes more essential. Importantly, the analytical bound of the expected LoP is always higher than the value of the actual LoP.

The second experiment was focused on the impact of number of nodes participating in the algorithm on LoP. The result of this experiment is presented in Figure 4. For each

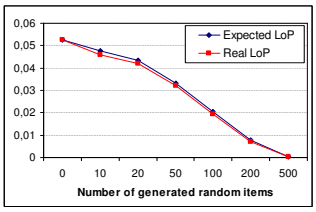

Fig. 3. LoP for set exposure vs. number of generated random items

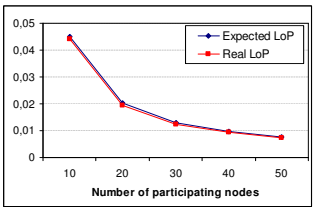

Fig. 4. LoP for set exposure vs. number of participating nodes

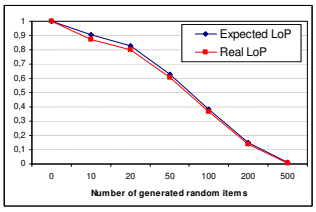

Fig. 5. Worst case LoP for set exposure vs. number of generated random items

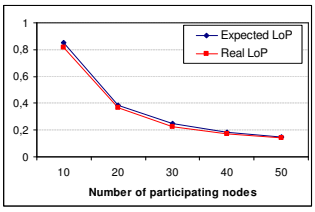

Fig. 6. Worst case LoP for set exposure vs. number of participating nodes

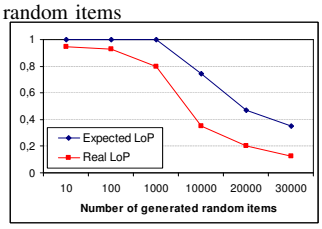

Fig. 7. Worst case LoP for item exposure vs. number of generated random items

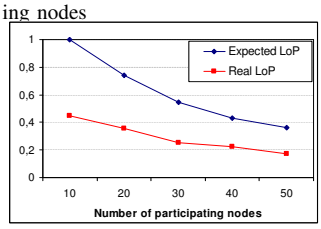

Fig. 8. Worst case LoP for item exposure vs. number of participating nodes

tested n we generated 100 random items by each node. As expected, both expected and actual LoP decrease while n increases. Again, the value of actual LoP is always lower than the analytical bound.

Finally, we also tested the worst case scenario (i.e. node 2 being aware that node 1 is the starting node). We present the results in Figures 5 and 6. The LoP is higher than the average case LoP. Nevertheless, generating random items or increasing number of nodes reduces the LoP significantly.

Item exposure. The second part of the experimental evaluation was focused on item exposure attacks. We are focusing on the worst case scenario (assuming that an adversary correctly identifies the fact that his predecessor is a leader). Recall from Equation 15 that the expected LoP of the protocol is equal to 0. LoP measure for the worst case was defined in Equation 14. Figure 7 presents the value of expected and actual LoP metrics as a function of the number of generated random items. It is worth noting that the value of actual LoP metric is around half of the value of expected LoP. Such a phenomenon is worth an explanation. When we derived Equation 17, we have assumed worst case scenario and assumed that $|x_1 \cap (r \cap (x - x_2))| = |x_1|$. On the other hand, in most cases the value of $|x_1 \cap (r \cap (x - x_2))|$ will be much smaller.

The second experiment for item exposure attacks measured the impact of number of participating nodes on the LoP value. For each tested value n we have generated 10,000 random items (which is 10% of the size of the domain). The result is presented in Figure 8. Similar to the previous case, an increase in the number of participants leads to a reduction in the value of expected and actual LoP.

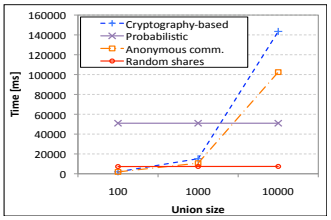

Fig. 9. Comparison of secure union protocols ($m = 2^{20}$)

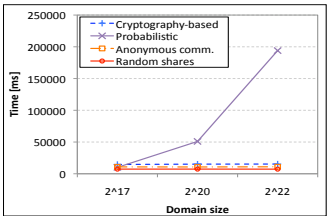

Fig. 10. Comparison of secure union protocols ($|result| = 1000$)

B. Cost of Secure Union Protocols

To compare the cost of all the secure union protocols, we have implemented the circuit-based, cryptography-based, probabilistic, as well as the anonymous communication-based protocols we described earlier. The circuit-based protocol was implemented using the FairplayMP [6] framework. The implementation of the cryptography-based protocol was based on RSA cipher. For the anonymous communication protocol, we used a communication circuit of 4 machines (excluding the sender and recipient).

We simulated a distributed environment with $n = 20$ nodes[1] and measured time of execution for each of the protocols except the circuit-based protocol. Due to a large time of execution, the runtime of the circuit-based protocol was estimated based on a performance analysis of the FairplayMP presented in [6]. We ran each of the other protocols for different domain sizes ($m = 2^{17}$, $m = 2^{20}$ and $m = 2^{22}$ items) and for different result size (100, 1000 and 10000 items). In the random shares protocol we set the size of the generated random set at each node to 10000 items.

The results are presented in Figures 9 and 10. First, we can observe that the cryptography-based, anonymous communication-based and random shares-based protocols do not depend significantly on size of the domain. On the other hand, for the probabilistic protocol, the runtime is strongly determined by this size. For $m = 2^{22}$ the protocol runtime is significantly larger than the random shares-based protocol even though the security provided by our solution is better. Finally, the costs of cryptography-based and anonymous communication-based protocols increase as the result size increases. This is the result of both protocols depending on an encryption. For a small result size, these protocols perform better than the random shares protocol. However, for larger result size, the random shares protocol performs much better.

Runtime of the circuit-based protocol was not placed on the plots due to very large values. For the domains we tested, FairplayMP generated circuits of size $2.8 * 10^7$, $2.2 * 10^8$ and $9 * 10^8$ gates. The estimated runtime for such circuits is 15 days, 127 days and 1.4 years, respectively. This makes the protocol impractical for most real life problems.

[1]The implementation of all the protocols in the simulated environment we used for evaluations can be downloaded from http://www.mathcs.emory.edu/Research/Area/datainfo/dobjects/sec

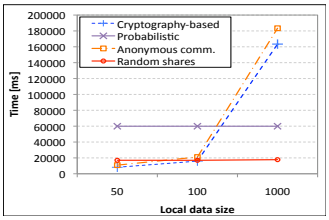

Fig. 11. Comparison of secure intersection protocols ($m = 2^{20}$)

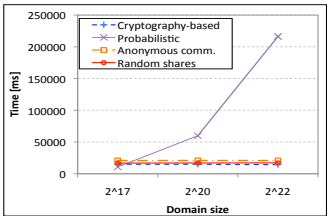

Fig. 12. Comparison of union protocols ($local\ data\ size = 100$)

C. Cost of Secure Intersection Protocols

Similar to the union protocols, we have implemented all the secure intersection protocols that we have discussed earlier. We simulated a distributed environment of n nodes. The random shares secure union protocol used in the intersection protocol generated 10000 random items by each node. The experiments were run for three different domain sizes, $m = 2^{17}$, $m = 2^{20}$ and $m = 2^{22}$ items. We also tested different input size at each node (100, 1000 and 10000 items). Note that this is different from the secure union experiments which tested different result size. This is due to the fact that the result of an intersection tends to be only a small fraction of the nodes' input data.

The results are presented in Figures 11 and 12. It can be observed that the results are quite similar to those of secure union protocol. As expected, the cryptography-based, anonymous communication-based and random shares-based protocols do not depend significantly on the domain size and the runtime of probabilistic protocol is strongly determined by this size. The costs of cryptography-based and anonymous communication-based protocols increase as the result size increases.

IX. Conclusion

In this paper we have discussed different secure union and intersection protocols that have been proposed in a literature. We have analyzed and evaluated the cost and feasibility of each of the approaches. In terms of the cost, while the circuit-based and probabilistic protocols turned out to be too costly for larger domain size, the cryptography and anonymous communication-based approaches performed quite well. However, for larger result size even these protocols have quite significant overhead.

We also presented a simple and intuitive secure union and intersection protocol based on the random shares approach. We have analyzed the security of our approach including both, regular attacks and collusion between nodes. Our protocols guarantee desired level of security that can be adjusted by users and incurs reasonable cost. We believe that it is desirable to make a tradeoff between security, efficiency and possibly accuracy for certain practical settings.

Acknowledgements

We thank Prof. Elisa Bertino for pointing us to anonymous communication protocols as a potential solution for the problem.

References

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *J. Cryptol.*, 2(1), 1990.
[2] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth ranked element, 2004.
[3] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases, 2003.
[4] R. Agrawal and E. Terzi. On honesty in sovereign information sharing. In *EDBT*, pages 240–256, 2006.
[5] M. Bawa, R. J. Bayardo, Jr., and R. Agrawal. Privacy-preserving indexing of documents on the network. In *VLDB*, 2003.
[6] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp: a system for secure multi-party computation. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, 2008.
[7] S. Böttcher and S. Obermeier. Secure set union and bag union computation for guaranteeing anonymity of distrustful participants. *JSW*, 3(1):9–17, 2008.
[8] C. Clifton, M. Kantarcioglu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining, 2003.
[9] L. K. Dawn and D. Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005, LNCS*, pages 241–257. Springer, 2005.
[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.
[11] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *NSPW '01: Proceedings of the 2001 workshop on New security paradigms*, 2001.
[12] W. Du and Z. Zhan. Building decision tree classifier on private data. In *CRPIT '14: Proceedings of the IEEE international conference on Privacy, security and data mining*, 2002.
[13] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.
[14] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.
[15] W. Jiang, C. Clifton, and M. Kantarciouglu. Transforming semi-honest protocols to ensure accountability. *Data Knowl. Eng.*, 65(1), 2008.
[16] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(9), 2004.
[17] H. Kargupta, K. Das, and K. Liu. Multi-party, privacy-preserving distributed data mining using a game theoretic framework. In *PKDD 2007: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.
[18] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3), 2002.
[19] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197, 2008. http://eprint.iacr.org/.
[20] B. Malin and L. Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3):179–192, 2004.
[21] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd edition, 1996.
[22] Y. Tao, X. Xiao, J. Li, and D. Zhang. On anti-corruption privacy preserving publication. *ICDE*, 2008.
[23] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *SIGKDD*, 2002.
[24] J. Vaidya and C. Clifton. Privacy-preserving data mining: Why, how, and when. *IEEE Security & Privacy*, 2(6):19–27, 2004.
[25] J. Vaidya, M. Kantarcioglu, and C. Clifton. Privacy-preserving naïve bayes classification. *VLDB J.*, 17(4):879–898, 2008.
[26] L. Xiong, S. Chitti, and L. Liu. Preserving data privacy for outsourcing data aggregation services. *ACM Transactions on Internet Technology (TOIT)*, 7(3), 2007.
[27] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SIAM SDM*, 2005.
[28] A. C.-C. Yao. How to generate and exchange secrets. In *SFCS '86: Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.
[29] N. Zhang and W. Zhao. Distributed privacy preserving information sharing. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 889–900. VLDB Endowment, 2005.