Privacy-Preserving Network Aggregation

Troy Raeder[1,2], Marina Blanton[2], Nitesh V. Chawla[1,2], and Keith Frikken[3]

[1] Interdisciplinary Center for Network Science and Applications
University of Notre Dame, Notre Dame IN 46556 USA
{traeder, nchawla}@cse.nd.edu
[2] Department of Computer Science and Engineering
University of Notre Dame, Notre Dame IN 46556 USA
mblanton@cse.nd.edu
[3] Miami University, Oxford OH 45056 USA
frikkekb@muohio.edu

Abstract. Consider the scenario where information about a large network is distributed across several different parties or commercial entities. Intuitively, we would expect that the *aggregate* network formed by combining the individual private networks would be a more faithful representation of the network phenomenon as a whole. However, privacy preservation of the individual networks becomes a mandate. Thus, it would be useful, given several portions of an underlying network $p_1 \dots p_n$, to securely compute the aggregate of all the networks p_i in a manner such that no party learns information about any other party's network. In this work, we propose a novel privacy preservation protocol for the non-trivial case of weighted networks. The protocol is secure against malicious adversaries.

1 Introduction

As the collection of social network data by enterprises increases, so too does the incidence of proprietary network data. One simple example arises from so-called "viral marketing" campaigns, which exploit the social networks of individuals for targeted marketing. In such a program, a new product is distributed to a few (hopefully influential) individuals with the goal that friends will then buy the product on their recommendation. If the success of recommendations can be tracked, future campaigns can take advantage of the previously-collected network information by targeting "influential" viral marketers, as identified by network structure.

Another scenario comes from the analysis of transactional data as a network of products, where the edge between two products p_i and p_j has a weight w_{ij} equal to the number of times p_i and p_j are purchased together. It has been shown that the structure of such a network can be exploited in order to discover meaningful complex relationships between products and propose potentially profitable promotions [13].

In both cases outlined above, any one view of the network may be *biased* [1, 7], meaning that it is an incomplete or unfaithful representation of the underlying "true" relationships. The simplest way to overcome such a bias is to combine networks from multiple sources on the principle that two networks are unlikely to suffer the same bias, or that the combination of networks will offer a more complete view of the true network.

When networks are highly proprietary or private, participants may not be comfortable disclosing their networks to each other. For instance, an aggregate social network may be useful in collecting product recommendations (if you trust your friends and their friends more than the general population), but the structure of individual social networks is often considered private. In the product networks scenario, while different stores may benefit from sharing, they would still like to preserve their proprietary information such as exact sales and product information. It is with this motivation that we discuss secure methods for network aggregation. Specifically, given a series of networks $\mathcal{G}_1 = (V_1, E_1), \ldots, \mathcal{G}_n = (V_n, E_n)$ from participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$, we show how to produce an aggregate network $\mathcal{G}$ in a manner such that no participant learns anything about another participant's network. **We then develop a novel protocol for the case of weighted networks that is secure in the presence of malicious adversaries.**

In this paper, we use the product networks scenario of [13] as the driving scenario. Specifically, we develop a protocol with which a set of n stores, selling ℓ products between them, participate in joint computation to securely determine c_{jk}, the number of times product j and product k have sold together in all stores combined (without revealing any information about the products that any one store sells). Each store chooses the products about which it would like to learn, and the stores jointly compute the c_{jk} without leaking unintended information to any of the participants. That is, each store only learns the total counts for products of its interest and other stores do not learn any of the inputs. In addition to the full protocol, we present strategies for efficient implementation and timing results based on real-world data.

It is worth noting that the protocol developed is general and can be applied to any application requiring a secure aggregation of networks. Our focus on product networks is driven by our ability to conduct empirical experiments, since we have data available.

The remainder of the paper is organized as follows: Section 2 describes the problem of secure network aggregation in general and outlines the difficulties associated with weighted networks. Section 3 describes our protocol for secure aggregation of weighted networks, makes suggestions for efficient implementation, and provides timing results on real-world data. Finally, Section 4 offers concluding remarks.

2 Secure Network Aggregation

Assume a series of participants $\mathcal{P}_1 \ldots \mathcal{P}_n$ wish to combine their private networks $\mathcal{G}_1 = (V_1, E_1) \ldots \mathcal{G}_n = (V_n, E_n)$. These networks can be either weighted or unweighted, but we assume that if any network has edge weights, all do.

In the case of unweighted networks, we can define an aggregate network as the "union" of individual constituent networks: a vertex or edge appears in the aggregate network if it appears in any individual network. In this case, aggregation is simply a union of the sets of vertices and edges, where edges are ordered pairs of vertices. Protocols exist [6] to compute this union efficiently and securely even in the presence of malicious adversaries. Alternatively, one could specify that a vertex or edge appears in the aggregate network only if it appears in at least some number k of individual networks. This is simply a formulation of the *secure over-threshold set union* problem, which has been solved in [9] for the case of semi-honest adversaries.

The case of weighted networks is more difficult. There is no obviously correct way to aggregate edge weights (one could take the minimum, maximum, sum, average, or any other quantity) and no existing secure protocols permit the combination of information such as edge weights during the computation of a set union. If we assume that the desired aggregate is "sum", as in the case of the product networks mentioned above, one could adapt a secure association-rules protocol (i.e. [8, 15]) to our problem in the following manner: For each edge (a, b) of weight w, produce w copies of the transaction `{a, b}`. Then run a secure association rules protocol with support s and zero confidence. The two-item association rules in the result will be the edges in the aggregate network whose total weight is at least s.

There are several problems with this approach. First, the size of the transaction database (and therefore the complexity of the algorithm) will depend on the edge weights, which may be arbitrarily large. Second, in these protocols, one participant learns the answer and must forward it to all others rather than informing all participants simultaneously. Third, these protocols do not preserve the actual aggregate (sum), but merely whether the sum exceeds a threshold. Our approach, which we present in the next section, addresses all these concerns.

Furthermore, we provide a mechanism whereby participants commit upfront to the vertices (i.e. products or individuals) about which they would like to learn. This commitment, which can be omitted for efficiency if deemed unnecessary, provides added privacy in that participants can only learn about what they already know. In a viral marketing scenario, for example, participant $\mathcal{P}_i$ could obtain more complete neighborhood information about individuals already in their network but would learn nothing about individuals in $\mathcal{P}_j$'s network that $\mathcal{P}_i$ had never encountered. Thus $\mathcal{P}_j$'s competitive advantage is sustained.

3 Privacy-Preserving Solution

In this section, after providing background information, we present our solution, which we call Private Product Correlation (PPC) protocol. We analyze its complexity, propose several efficiency improvements and demonstrate performance on real data. Due to space constraints, the formal security proof has been deferred to [12].

3.1 Preliminaries

Homomorphic encryption Our solution utilizes semantically secure homomorphic encryption that allows computation on encrypted data without knowledge of the corresponding plaintext. In particular, we use public-key additively homomorphic encryption such as Paillier [11]. Suppose there is a public-private key pair (pk, sk); we denote encryption of message m as $E_{pk}(m)$ and decryption of ciphertext c as $D_{sk}(c)$. The additive property gives us $D_{sk}(E_{pk}(m_1) \cdot E_{pk}(m_2)) = m_1 + m_2$ and $E_{pk}(m)^a = E_{pk}(m \cdot a)$. It is also possible, given a ciphertext $c = E_{pk}(m)$, to compute a re-encryption of m such that it is not feasible to tell whether the two ciphertexts correspond to the same message or not; this is done by multiplying the ciphertext by an encryption of 0.

Our protocols use a threshold version of homomorphic encryption. In an (n, k)-threshold encryption scheme, the decryption key is distributed among n parties and the participation of k of them ($k \leq n$) is required to decrypt a ciphertext.

Zero-knowledge proofs Our protocols rely on zero-knowledge proofs of knowledge from prior literature. In particular, we use a proof of plaintext multiplication defined as follows: given ciphertexts $c_1 = E_{pk}(a)$, $c_2 = E_{pk}(b)$, and $c_3 = E_{pk}(c)$, the prover proves that c corresponds to multiplication of a and b, i.e., $c = a \cdot b$. Cramer et al. [2] give a zero-knowledge protocol for this proof using Paillier homomorphic encryption, which is the type of encryption used in this work.

Privacy-preserving set operations Prior literature [5, 6, 9] contains results that permit privacy-preserving operations on sets (or multi-sets). A set $S = \{s_1, s_2, \ldots, s_\ell\}$ is represented as the polynomial $f_S(x) = (x - s_1)(x - s_2) \cdots (x - s_\ell)$. This representation has the property that $f_S(s) = 0$ if and only if $s \in S$.

Privacy-preserving operations on sets use encrypted representations of sets. Given a polynomial $f(x) = a_\ell x^\ell + a_{\ell-1}x^{\ell-1} + \ldots + a_1 x + a_0$, its encryption is formed as encryption of each coefficient a_i: $E_{pk}(f) = (E_{pk}(a_\ell), \ldots, E_{pk}(a_0))$. This representation can be used to perform set operations in privacy-preserving manner. One such an operation used in our solution is *polynomial evaluation*, which given $E_{pk}(f)$, y, and public parameters allows one to compute $E_{pk}(f(y))$. This is done by computing the product $\prod_{i=0}^{\ell} E_{pk}(a_i)^{y^i}$. We also utilize the *set union* protocol of [6], which is the fastest protocol for computing the union of two sets.

3.2 Private Product Correlation Protocol

In our solution we assume that there are n participants (i.e., stores) $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Each participant $\mathcal{P}_i$ sells a number of products to which we refer as L_i. We assume that a unique naming convention is used, and different participants will use the same name for a particular product.

Overview of the solution A natural solution to the product correlation problem in a non-private setting proceeds as follows: each participant counts the number of instances two products were sold together at its store, across all pairs of products the participant offers. Given these counts, the aggregate counts are computed for each pair of products the participants collectively carry. Each participant then saves the aggregate counts corresponding to the products it is interested in. The same logic could be used in constructing a privacy-preserving protocol for product correlation: each participant computes the counts privately, all of them then engage in a variant of a set union protocol preserving (and summing) the counts during the protocol, and finally each participant performs a set intersection on the result to recover the counts for the products of interest.

The existing techniques do not allow this functionality to be implemented in the above form. That is, while privacy-preserving protocols for both set union and set intersection exist, they are not composable, i.e., they cannot be used as sub-protocols in a larger solution which is required to be secure. Furthermore, the way sets are represented in these protocols does not permit additional information (such as a count) to be stored with an element of the set. These limitations led us to design alternative mechanisms for achieving the above task.

Our protocol first requires that the participants agree on an (n, n)-threshold homomorphic encryption scheme and a naming convention for vertices. The simplest choice would be some sort of hashed unique identifier, such as UPC codes for product networks, or e-mail addresses, social security numbers, or ID codes in the more general case. Then, every participant commits to the set of products about which it would like to learn without revealing this set to others. Next, we employ a secure set-union protocol to determine the set of products on which we need to compute. Each participant prepares counts for all pairs of products in the set union and broadcasts encrypted counts to others. The participants jointly add the counts using the homomorphic properties of the encryption scheme. To allow a participant to learn information about the products to which he or she committed (without others learning anything), all parties will aid with the decryption of necessary (unknown to others) counts.

It is conceivably possible to construct a simpler protocol to achieve similar aims. One could repeatedly apply a secure sum protocol to compute the counts we desire, or could omit the commitment step and simply have participants learn about all pairs of products. However, we would argue that these solutions lack potentially desirable security properties. Our full protocol provides added security by preventing participants from learning an arbitrary amount of information regarding products other stores stock (which could result in stores refusing to participate). That is, by limiting the number of products about which a participant learns, we provide the stores with a useful utility without giving anyone the ability to abuse the knowledge they gain.

Protocol description The participants agree on a (n, n)-threshold homomorphic encryption scheme and generate a public-private key pair (pk, sk) for it.

PPC Protocol:

1. Each participant $\mathcal{P}_i$ creates a list of products $D_i = \{d_1, \ldots, d_{m_i}\}$ about which it would like to learn. $\mathcal{P}_i$ commits to this set by committing to the polynomial $Q_i(x) = (x - d_1) \cdots (x - d_{m_i}) = q_{m_i} x^{m_i} + q_{m_i - 1} x^{m_i - 1} + \cdots + q_1 x + q_0$ for constants $q_{m_i}, \ldots, q_0$. (Q_i may be padded with fake items to hide the size of the set D_i). More specifically, $\mathcal{P}_i$ posts $E_{pk}(q_{m_i}), \ldots, E_{pk}(q_0)$ along with a zero-knowledge proof that q_{m_i} is non-zero as described in sub-protocol NZProof below. The proof is necessary to provide a bound on the size of m_i.
2. Each participant $\mathcal{P}_i$ prepares a list of its products L_i. The participants engage in a privacy-preserving set union protocol to determine the set of products all of them sell. Let $L = \{p_1, \ldots, p_\ell\}$ denote the outcome of the protocol.
3. For each pair of products $p_j, p_k \in L$, $\mathcal{P}_i$ computes $E_{pk}(c^i_{jk})$, where c^i_{jk} is its count for the number of times products p_j and p_k were sold together for $\mathcal{P}_i$. Note that if at least one of p_j and p_k is not in L_i, c^i_{jk} will be 0. $\mathcal{P}_i$ broadcasts the values $E_{pk}(c^i_{jk})$ for each $0 \leq j, k \leq \ell$ $(j \neq k)$.
4. Each $\mathcal{P}_i$ locally computes the encryption of the sum of all counts for each product pair p_i, p_k as $E_{pk}(c_{jk}) = \prod_{i=1}^{n} E_{pk}(c^i_{jk}))$. $\mathcal{P}_i$ then rearranges the values to form tuples $(p_j, E_{pk}(c_{j1}), \ldots, E_{pk}(c_{j\ell}))$ for each $p_j \in L$.
5. Now each $\mathcal{P}_i$ obtains the decryption of counts of the products to which $\mathcal{P}_i$ committed in step 1 (i.e., all counts c_{jk} such that $p_j \in D_i$, $1 \leq k \leq \ell$, and $k \neq j$). To accomplish this, perform the following in parallel for each $\mathcal{P}_i$:

(a) One party posts $E_{pk}(Q_i(p_1)),\ E_{pk}(Q_i(p_2)),\ \ldots,\ E_{pk}(Q_i(p_\ell))$. Note that $E_{pk}(Q_i(p_j)) = E_{pk}(q_{m_i})^{p_j^{m_i}} \cdot E_{pk}(q_{m_i-1})^{p_j^{m_i-1}} \cdots E_{pk}(q_0)$, and since this is a deterministic process, everyone can verify the result of the computation. Also, note that $Q_i(p_j)$ will be 0 iff p_j was in D_i.

(b) For each value $E_{pk}(Q_i(p_j))$, $j = 1, \ldots, \ell$, the participants randomize the underlying plaintext by engaging in a NZRM (non-zero random multiplication) protocol described below (each participant executes the NZRM protocol in order). In this protocol each participant multiplies each plaintext by a random non-zero value and proves correctness of the computation. We denote the result of the computation by $E_{pk}(b^i_j)$. Note that b^i_j is 0 if $p_j \in D_i$ and a random value otherwise.

(c) For each $0 \leq j, k \leq \ell$ ($j \neq k$), one party computes the values $E_{pk}(b^i_j) \cdot E_{pk}(c_{jk}) = E_{pk}(b^i_j + c_{jk})$. Note that the encrypted value is c_{jk} if $p_j \in D_i$ and is a random value otherwise.

(d) The parties engage in a joint decryption protocol to reveal the values $b^i_j + c_{jk}$ to the $\mathcal{P}_i$ for all $0 \leq j, k \leq \ell$.

Sub-protocols NZProof Protocol: A user has a ciphertext c and would like to prove that the corresponding plaintext a (where $c = E_{pk}(a)$) is non-zero.

1. The prover chooses a random value b and posts $E_{pk}(b)$ and $E_{pk}(ab)$.
2. The prover proves in zero knowledge that the decryption of $E_{pk}(ab)$ corresponds to the multiplication of the decryptions of $E_{pk}(a)$ and $E_{pk}(b)$.
3. The prover decrypts $E_{pk}(ab)$ and posts ab (this value is jointly decrypted in case of threshold encryption). If ab is non-zero, so must a.

NZRM Protocol: Given a ciphertext $c = E_{pk}(a)$, a participant multiplies the underlying plaintext by a random non-zero value b, outputs $c' = E_{pk}(ab)$ and proves correctness of the computation.

1. The participant chooses a random value b and posts $E_{pk}(b)$ and $c' = E_{pk}(ab)$.
2. The participant proves in zero knowledge that the decryption of $E_{pk}(ab)$ corresponds to the multiplication of the decryptions of $E_{pk}(a)$ and $E_{pk}(b)$.
3. The participant also proves that $E_{pk}(b)$ encrypts a non-zero value using the NZProof protocol.

Complexity Analysis To simplify the analysis, let m be the upper bound on the number of items to which a participant commits (i.e., $m = \max_i\{m_i\}$). The total work and communication for participant $\mathcal{P}_i$ in step 1 of the protocol is $O(m+n)$, which amounts to $O(mn+n^2)$ communication across all parties. The computation and communication associated with the proof of nonzero q_{m_i} is constant with respect to m and n and does not affect the complexity. In step 2, the overhead associated with the semi-honest (malicious) version of the set union protocol is bounded by $O(n\ell)$ ($O(n\ell^2 + n^2\ell)$, resp.) computation and communication per person and therefore $O(n^2\ell)$ ($O(n^2\ell^2 + n^3\ell)$, resp.) overall communication. In step 3, each participant's work and communication is $O(\ell^2)$ resulting in $O(n\ell^2)$ overall communication. Step 4 involves $O(\ell^2)$ cheaper operations (modular multiplications) per participant and no communication.

To determine the output for a single participant $\mathcal{P}_i$ in step 5, the overhead is as follows: step 5a requires $O(m\ell)$ operations and the same amount of overall communication. Step 5b requires $O(\ell)$ work and communication per participant, resulting in the total of $O(n\ell)$ communication. Here the work and communication added by the $\mathsf{NZProof}$ protocol is constant per participant and per product, so the additional complexity introduced is $O(n\ell)$, which does not affect the asymptotic running time. Step 5c involves $O(\ell^2)$ computation and overall communication. Finally, step 5d involves $O(\ell^2)$ threshold decryptions, which amounts to $O(\ell^2)$ work and communication per participant resulting in $O(n\ell^2)$ total communication. Since this part is executed for each participant, the total communication of step 5 over all participants is $O(n^2\ell^2)$.

Thus, if the protocol is built to resist malicious adversaries, the work per participant is $O(n\ell^2 + n^2\ell)$ and the overall communication is $O(n^2\ell^2 + n^3\ell)$.

3.3 Efficiency Improvements

Polynomial multiplication and evaluation As described in [5], polynomial evaluation can be performed more efficiently by applying Horner's rule. Recall from section 3.1 that computing $E_{pk}(f(y))$ amounts to calculating $\prod_{i=0}^{\ell} E_{pk}(a_i)^{y^i}$. More efficient computation can be performed by evaluating it from "the inside out" as $E_{pk}(f(y)) = ((\cdots(E_{pk}(a_\ell)^y E_{pk}(a_{\ell-1}))^y \cdots)^y E_{pk}(a_1))^y E_{pk}(a_0)$. Considering that the polynomial is always evaluated on small values (compared to the size of the encryption modulus), this results in a significant performance improvement.

The set union protocol we utilize [6] also uses polynomial multiplication, which for large polynomials becomes inefficient. Given an encrypted polynomial $E_{pk}(f_1) = (E_{pk}(a_{\ell_1}), \ldots, E_{pk}(a_0))$ and another polynomial $f_2(x) = b_{\ell_2}x^{\ell_2} + \cdots + b_0$, the multiplication consists of computing (encrypted) coefficients c_i of their product: $E_{pk}(c_i) = \prod_{j=\max\{0,i-\ell_2\}}^{\min\{i,\ell_1\}} E_{pk}(a_j)^{b_{i-j}}$ for $i = 0, \ldots, \ell_1 + \ell_2$. This can be performed faster by using multi-base exponentiation (see, e.g., [10]), where instead of computing exponentiations $g_1^{x_1}, \ldots, g_k^{x_k}$ separately, exponentiation is performed simultaneously as $g_1^{x_1} \cdots g_k^{x_k}$ for a fixed (small) value of k. This can speed up computation by several times.

Packing Assume that the (total) c_{jk} values are at most 2^M. It is likely that $M \ll \rho$, where ρ is the number of bits in the modulus of the encryption scheme. In Step 3 of the PPC protocol, a participant posts ℓ encryptions. We can reduce this number by storing $s = \lfloor \frac{\rho}{M} \rfloor$ values in a single encryption, which would require only $\lceil \frac{\ell}{s} \rceil$ encryptions.

To do the compression, suppose we want to place M-bit values $x_1, ..., x_s$ into a single encryption. We then compute $E_{pk}(\sum_{i=1}^{s} 2^{M(i-1)}x_i)$. Note that as long as individual results do not get larger than M bits, we can add to such compressed encryptions (to obtain the value representing the pairwise values) and we can multiply them by constants. In the protocol, addition is the only operation performed on the counts. Such compressed encryptions of counts can also easily be used in Step 5 of the protocol if only the counts that correspond to the same product are combined together (e.g., c_{jk} and $c_{jk'}$ can be included in the same ciphertext for any k, k'). Doing this compression does not reduce the asymptotic communication of the protocol (as this has no effect on the set union), but it does improve the performance of Steps 3–5 in the protocol.

Leaking products with zero sales In our data, there are many products that never sell together, either because they are relatively unpopular or because they are not available at the same time. If this information is not valuable to an adversary, pairs of products with zero sales can be excluded from the protocol, meaning that their counts do not need to be encrypted, combined, or decrypted.

3.4 Performance

One persistent concern with privacy-preserving data mining protocols is that they tend to be unacceptably computation-intensive. We now present a discussion of the performance of our protocol and the optimizations that were required to make it tractable in practice. Such a discussion serves both to demonstrate the feasibility of our algorithm and to serve as a baseline for evaluating privacy-preserving data mining techniques.

We implemented the protocol in C using (n,n)-threshold Paillier encryption [2–4] as the encryption scheme and the protocol of [6] for set union. For efficiency, we extend the set union protocol as follows: the participants decide on a number, B, of buckets, and each participant divides its products among the B buckets according to a hash function. The participants run the set union protocol B times and combine the results of the runs. This can be accomplished without leaking additional information since the combination of several unions is the same as the final union. If the participants split the products uniformly among $B = \frac{\ell}{\log(\ell)}$ buckets, the buckets will contain $\log(\ell)$ products on average, reducing the time complexity of the set union from $O(n\ell^2)$ to $O(n\ell\log(\ell))$. In practice, the participants do not know the size of the union beforehand, so we approximated ℓ with $n \cdot \ell_i$ where ℓ_i is the number of products in each store. We find that this significantly reduces the running time of the set union.

In conducting the experiments, we accounted for the parallelism afforded by the protocol (each participant doing computation in parallel). However, in situations where the computation must be serialized we time the computation of all participants. Specifically we do not include key generation, the construction of unencrypted polynomials for the set union, or the construction of the commitment polynomials. These are pre-processing steps outside the context of PPC. For steps that can be done by all participants simultaneously (steps 2, 3, 4, 5d), we only time the effort of one participant. In steps 4 and 5d, the effort to combine results is also timed.

All experiments were done with a 1024-bit key in keeping with modern standards for public-key encryption, and are run on real-world transaction data from a convenience store at the University of Notre Dame. For the purposes of the experiment, we constructed several stores from a single dataset by randomly assigning products to each store from a pre-determined list of top-selling products. The size of the set union depends on how much overlap there was between the selections of the different stores and is, therefore, random.

Table 1 shows performance as a function of the number of participants n, the size of the set union ℓ, and the size of each store. We present timing results for (i) the full protocol carried out on all pairs of products and (ii) when pairs of products that were never sold together were excluded, as mentioned in Section 3.3. We also report the speedup obtained from the second scenario over the first. The time taken to complete the protocol in both cases scales most significantly with the size of the set union. While increases in the number of participants has some effect, it is partially offset by the increase in the amount of available parallelism afforded by the participation of multiple entities.

Table 1. Timing results for protocol execution.

Participants	Products Per Participant	Products in Union	Time (all pairs)	Time (no zeros)	Speedup
3	100	239	105:25	14:21	7.35
3	200	461	386:14	45:40	8.46
3	400	942	1,637:09	98:42	16.69
5	100	312	293:33	61:57	4.74
5	200	601	1,097:21	167:49	6.54
5	400	1231	4,794:07	338:49	14.14
10	100	375	939:17	336:15	2.79
10	200	744	1,883:04	844:42	2.22
10	400	1502	7,191:15	1506:02	4.77

Improvement as a result of leaking zero counts is substantial, achieving at least 2.22 times speedup. In general, the benefit to leaking zeros increases with the size of the union as zeros become more likely and decreases with the number of participants, as the overhead of each computation is larger. The only exception was the case with 10 stores and 200 products, in which we observe a decrease in speedup from 2.79 to 2.22. Based on other results, this seems to be an aberration, perhaps caused by unusual activity on the machine running the experiment. In all, we observe that leaking zero counts makes the computation substantially more tractable. Whereas computing all O(ℓ^2) counts takes almost five days with ten participants and 1500 products in the union, leaking zero counts reduces computation time to just over one day.

4 Conclusion

The goal of this work was to design solutions that utilize the power of networks while ensuring that privacy of the affected parties is preserved and no unintended information leakage takes place. We considered a family of product networks where the stores want to share information about their product/item networks to form a global network, such that they can query this global network for efficient marketing and pricing. The collection of large volumes of customer transaction data is commonplace among both large and small retailers of consumer products. The data collected by any one retailer may potentially be *biased*, i.e., it does not accurately reflect the level of demand for products in the store, for any number of reasons. Moreover, competitive enterprises can mutually improve their standing in a market by sharing information with rivals [14].

To address these concerns, we developed the Private Product Correlation (PPC) protocol for the secure exchange of aggregate network information. The protocol provides more information to participants (namely, a sum of counts) and is secure against malicious as well as semi-honest adversaries. While our work targeted the product networks, it is general enough for applicability to any scenario that requires an aggregation of networks such that no participant learns anything about another participant's network.

We empirically demonstrate the efficacy of the developed protocol by presenting timing results, which show that our framework is tractable for participants with reasonable computing power. Our results suggest that execution of the protocol would take on the order of days for participants with a modest number of products using widely-available off-the-shelf hardware. Furthermore, if participants are comfortable with leaking pairs of products that they do not sell together at all, they will see significant gains in performance.

We hope that the above-described analytical techniques and privacy-preserving protocol lead to the increased availability of transactional datasets for scientific study. Product networks contain significantly less information than transaction databases, because some information is lost in the aggregation process. The combination of several product networks, then, should reveal almost nothing about one store's marketing model. As such, retailers could consider an aggregated product network to be devoid of proprietary information and may allow this information to be released and studied.

Acknowledgments This work was supported in part by the NET Institute, the Air Force Office of Scientific Research under grant AFOSR-FA9550-09-1-0223 and the National Science Foundation under grants CNS-0915843 and BCS-0826958.

References

1. N. V. Chawla and G. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *JAIR*, 23:331–366, 2005.
2. R. Cramer, I. Damgard, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT'01*, volume 2045 of *LNCS*, pages 280–299, 2001.
3. I. Damgard and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC*, pages 90–104, 2001.
4. P. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography*, volume 1962 of *LNCS*, pages 90–104, 2000.
5. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 1–19, 2004.
6. K. Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security (ACNS'07)*, volume 4521 of *LNCS*, pages 237–252, 2007.
7. J. Heckman. Sample selection bias as a specification error. *Econometrica: Journal of the econometric society*, pages 153–161, 1979.
8. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE TKDE*, 16(9):1026–1037, 2004.
9. L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO'05*, volume 3621 of *LNCS*, pages 241–257, 2005.
10. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. 1996.
11. P. Paillier. Public key cryptosystem based on composite degree residue classes. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, 1999.
12. T. Raeder, M. Blanton, N. Chawla, and K. Frikken. Privacy-preserving network aggregation. Technical Report TR-2010-02, Notre Dame Computer Science and Engineering, 2010.
13. T. Raeder and N. Chawla. Modeling a store's product space as a social network. In *Proceedings of ASONAM*, Athens, Greece, 2009.
14. M. Raith. A general model of information sharing in oligopoly. *Journal of Economic Theory*, 71(1):260–288, 1996.
15. J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of KDD*, pages 639–644, New York, NY, USA, 2002. ACM.