An Efficient Bloom Filter Based Solution for Multiparty Private Matching*

Pierre K.Y. Lai S.M. Yiu K.P. Chow C.F. Chong Lucas C.K. Hui

Department of Computer Science
The University of Hong Kong
Pokfulam Road
Hong Kong

Abstract - The issue of privacy becomes more and more important. On the other hand, online collaboration among different parties is almost unavoidable. How to allow collaboration while protecting one's privacy is an important problem and attracted a lot of attention. Unfortunately, most of the problems in this area still remain unsolved. In this paper, we investigate one of the fundamental problems called multiparty private matching in which N users want to compute the set of records that are present in everyone's database without revealing unnecessary information such as records not in others' databases or any intermediate resulting sets. Existing solutions are either inefficient or require a mediator. We propose a bloom-filter based solution, which does not require a trusted third party and is more efficient than all existing solutions. We also show that although bloom filter may return false positive, the probability of getting false positive can be set to very low that could still be useful to a lot of applications.

Keywords: **privacy, bloom filter, multiparty matching, database operation**

1 Introduction

It is common for autonomous entities to integrate their data for mutual benefits. Recently, the sharing of data has raised significant privacy concerns. In particular, none of the entities would like to reveal any information besides the information necessary for the integration. How this can be done is non-trivial and has attracted a lot of research. However, many related problems in this area remain unsolved. In this paper, we consider one of the fundamental problems, called the *private matching problem*. The problem is defined as follows. Two non-trusting parties wish to compute the common data elements over their private databases in such a way that no other information is revealed except the list of common elements. The problem can be extended to a multiparty setting over Internet, in which common elements are computed from private databases owned by multiple non-trusting parties. One trivial solution to this extension is performing private matching with pairs of two databases using any existing schemes for the two-party case. However, in addition to the fundamental privacy concern on private information of individual databases, one needs to take into account the disclosure of intermediate results for the multiparty case. This makes the simple extension not appropriate as it would either incur *N*-folded communication overhead or intermediate results are uncovered. None of the existing schemes are able to provide an efficient solution to solve this multiparty private matching problem without a trusted third party.

1.1 Our Contributions

We present a very efficient protocol for computing private matching in a multiparty setting. The scheme can provide an adequate level of privacy while enjoying a minimal communication overhead,

* This research is supported in part by a grant from the Research Grants Council of the HKSAR, China (Project No. HKU/7136/04E).

making it particularly favourable for applications where a large number of databases would be involved. Unlike many other schemes, our construction does not require the intervention of a trusted third party, which can hardly be available in real situations. Our solution is based on the concept of bloom filter. We believe that the solution proposed in this paper is practical and easy to put into practice in real scenarios. Note that bloom filter may return false positive, however, we show that probability of false positive can be set to very low. This problem has a lot of applications for online collaboration among autonomous entities. The following shows two general scenarios from which many other applications can be elaborated.

Uncovering disgraced identities
Assume that credit card companies maintain a list of customers who are likely to run into bad debt according to their transaction records. To minimize the loss due to uncollected loans, they want to take early actions to those highly potential bad-debtors. A person is labelled as a highly potential bad-debtor if his name appears on the lists of all these companies. Due to privacy concerns, these companies cannot share their plain lists with each other when compiling the list of highly potential bad-debtors.

Selective alliance among business counterparts
Suppose that there are multiple enterprises which are competitors providing similar services. In a special occasion, they want to expand their business by bringing up ad-hoc joint projects. They want to identify the items which are common in their interests while hiding other information in order to retain competitiveness.

1.2 Related Work

Privacy-preserving distributed computation has been an engaging research area in recent years. We review the works that address a similar problem as ours as follows. Secure computation of any functions in multiparty environment has been shown to be possible by [1], [2] and [3] for different scenarios. These protocols are similar in their constructions: representing the computed function as a circuit and evaluating it. They are proven to be secure but the massive computation overhead limits them to work with only a small amount of data.

The first work that addresses the private matching problem is [4]. Their best construction requires oblivious evaluation of n^2 linear polynomials. Again, it is too expensive for matching data in the context of databases. Subsequently, two other works [9] and [10] studied a similar problem introducing the framework for database uses. The latter is constructed based on the former with the addition of Data Ownership Certificate, hoping to address the data spoofing problem. The scheme requires some authentication information generated by the data owner to be stored with the data in these databases, making it inapplicable to many scenarios where data ownership can hardly be defined. Also, intermediate results are likely to be revealed when these works are put forward to the multiparty setting.

In [6], the authors propose a system for distributed join queries which is also based on the use of Bloom filters. However, their system is specialised for the context of clinical case research. Also, the requirement of a mediator, who is supposed to be trusted by everyone, sets it apart from our proposed schemes. A more recent work [5] proposes relatively efficient protocols with the use of homomorphic encryption and balanced hashing for two-party computation of set intersection. At the end of the paper, they sketched a protocol for the multiparty setting, which requires O(Nn) encrypt-decrypt-encrypt-decrypt operations for individual parties where N is the number of users and n is the number of records. Our protocols outperform theirs evidently in terms of efficiency.

1.3 Outline

The rest of the paper is organized as follows. Section 2 gives the notations for this paper and a brief review of Bloom filters. Section 3 presents the protocols for multiparty private matching. Section 4 gives the analysis of the proposed scheme in terms of efficiency, credibility and security. Section 5 concludes this paper.

2 Preliminaries

2.1 Notations

The notations for the rest of this paper are as follows.

B_i	the Bloom filter representing database A_i;
$B_{i,j}$	the j^{th} segment of the Bloom filter B_i;
$B_{ij}[k]$	the k^{th} bit of the Bloom filter segment $B_{i,j}$;

2.2 Bloom Filters

A Bloom filter [7] is a randomized data structure for representing a set $A = \{a_1, a_2,\ldots, a_n\}$ to encode set membership. Suppose that we have an m-bit array B, with all bits initialized to zero. We also have k independent hash functions $h_1, h_2,\ldots, h_k$ that take all inputs and produce outputs distributed uniformly over the range [1..m]. To include a record a into B, we compute $h_1(a), h_2(a),\ldots, h_k(a)$ and set the bits at these k positions to 1.

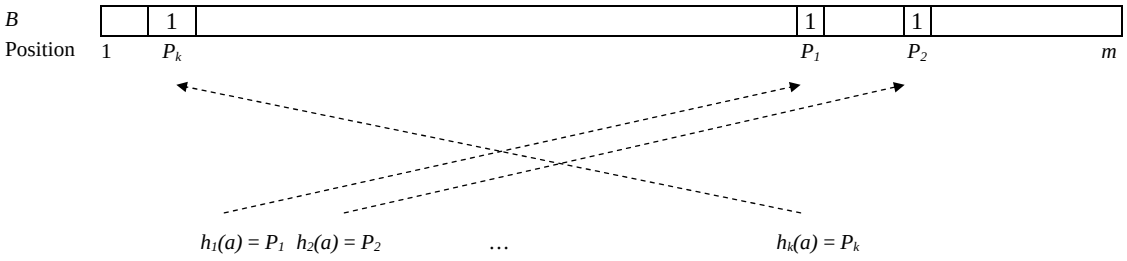

Figure 1

To answer a query for x, the positions at $h_1(x), h_2(x), \ldots, h_k(x)$ are examined. If any of them is zero, x is certainly not included in B. Bloom Filters provide space-efficient storage for this kind of membership queries at the cost of a probability of false positive. That is, even all the k bits associated with x are found to be ones, x may actually be absent in the set A. The false positive rates, however, are shown to be tuneable by careful selection of parameters.

3 Our Solution for Multiparty Private Matching

3.1 Framework

- *Code(R):* Take a record *R* and return the Bloom filter representing *R*.
- *Build(A):* Given a database *A*, return the union of *Code()* on each record as the result. The output is the Bloom filter representing *A*.
- *Partition(B)*: Given a Bloom filter *B*, partition it into *N* equal-length segments of *m/N* bits each.
- *Merge(S_1,S_2,…,S_N)*: On the input of *N* Bloom filters segments, do the following steps:
  1. Prepare a zero bit vector, *S*, of *m/N* bits.
  2. For $j = 1$ to m/N (i.e. each bit position)
     Set $S_{[j]} = S_{1[j]} \wedge S_{2[j]} \wedge \ldots \wedge S_{N[j]}$ where ^ is the bit-wise AND operation.
  3. Output *S*.

3.2 The Scheme

The general idea of the scheme is as follows. Firstly, each party prepares a representation of its own database *(Step 1 – Step2)*. After that, they exchange a part of their representation in the way that apparently each party gets an equal amount of information *(Step 3)*. On the possession of the partial information of other parties, one can screen out the unnecessary information so that only those contribute to the final result will be retained *(Step 4)*. Again, they exchange the partial results *(Step 5)* and compute the result on their own *(Step 6 – Step7)*.

The following shows the construction.
For each party P_i:

Step 1: Compute the Bloom filter of its database:
$Build(A_i) \rightarrow B_i$

Step 2: Partition the resulting Bloom filter into *N* even segments:
$Partition(B_i) \rightarrow B_{i,1}, B_{i,2}, \ldots, B_{i,N}$

Step 3: Exchange the bloom filter segments with other parties:
Send $B_{i,j}$ to P_j and receive $B_{j,i}$ from P_j $\quad \forall\ j \in [\,1..N\,]$ and $j \neq i$

Step 4: Compute the partial information from the received segments:
$Merge(B_{1,i} \ldots B_{N,i}) \rightarrow R_i$
where R_i is the i^{th} segment of the resulting bloom filter

Step 5: Exchange the partial results with other parties:
Send R_i to P_j and receive R_j from P_j $\quad \forall\ j \in [\,1..N\,]$ and $j \neq i$

Step 6: Obtain the resulting *R*:
$R = R_1 \,|R_2\,| \ldots |\, R_N$

Step 7: Check each record *r* against *R*:
If *R* contains *Code(r)*, include *r* in the result set

4 Analysis

In this section, we evaluate our proposed schemes in terms of credibility, efficiency and security. We look at these issues from the point of the use of Bloom filters in our schemes.

Credibility Analysis

The reliability of using Bloom filters for membership queries has been studied in many previous works [7, 8]. The general approach is to calculate the false positive rate *fpr*, which is the probability of having ones at all the *k* positions associated with *X* by chance. The *fpr* of a bloom filter is

$$\left[1-(1-\frac{1}{m})^{kn}\right]^{k} \qquad (\mathbf{1})$$

For a false positive to remain at the final resulting bloom filter, all the corresponding bits have to be set to ones by chance in each individual bloom filter. Therefore, the *fpr* is

$$\left\{\left[1-(1-\frac{1}{m})^{kn}\right]^{k}\right\}^{N} \qquad (\mathbf{2})$$

Since *fpr* would probably be a number much smaller than 1, powering it *N* times improves the accuracy to a considerable degree in our scheme.

Efficiency Analysis

The computation cost for each party is $n \times k \times C_h$ where n is the number of records in each database; k is the number of hash functions and C_h is the cost of computing a hash function. And the communication cost (in bits) for each pair of parties is *2m/N* where m is the size of a Bloom filter and N is the number of parties.

Our scheme is very efficient and can outperform other existing schemes evidently in terms of the incurred communication overhead, as the amount of data sent is independent of the number of records.

Security Analysis

The probability of guessing the existence of a record using the partial Bloom filter one received from a particular party:

$$\left[1-(1-\frac{1}{m})^{kn}\right]^{\frac{k}{N}} \qquad (3)$$

A high *fpr* value implies that it is unlikely for someone to make a guess correctly. Therefore, bigger *fpr* suggests a higher level of security. We explore the relationship between *fpr* and *N* using the following setup: *m* = 32KB; *n* = 10000; and *k* = 18. Figure 2 shows the estimated *fpr* for a partial Bloom filter against various *N* values. It suggests that our schemes produce high *fpr* such that the existence of a particular record is not certain based on the information available from a partial Bloom filter only.

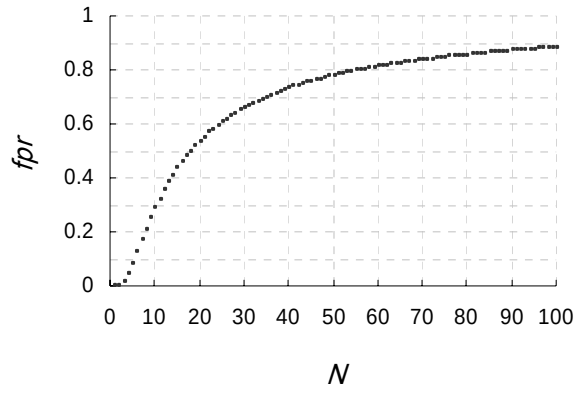

Figure 2

5 Conclusions

We develop a practical protocol for computing private matching in a multiparty setting. We show that our scheme is efficient and particularly applicable to situations where the number of involving parties is large. In such cases, the information available in a partial bloom filter is negligible while the communication overhead is still the same. We believe that our schemes are simple to understand and easy to be put into practice in real scenarios.

6 References

[1] O. Goldreich, S. Micali and A. Wigderson. “How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority”. Proceedings of the 19th Annual Symposium on the Theory of Computing (STOC), ACM, 1987, pp. 218-229.

[2] M. Ben-Or, S. Goldwasser and A. Wigderson. “Completeness theorems for non cryptographic fault tolerant distributed computation”. Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC), ACM, 1988, pp. 1–9.

[3] D. Chaum, C. Crepeau and I. Damgard. “Multiparty unconditionally secure protocols”. Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC), ACM, 1988, pp. 11–19.

[4] M. Naor and B. Pinkas. “Oblivious Transfer and Polynomial Evaluation”. In Proc. of the 31th ACM Symposium on Theory of Computing, pages 245–254, Atlanta, Georgia, 1999.

[5] M. Freedman, K. Nissim and B. Pinkas. “Efficient Private Matching and Set Intersection”. In Advances in Cryptology – Eurocrypt ’2004 Proceedings, LNCS 3027, Springer-Verlag, pp. 1-19, May 2004.

[6] G. Schadow, S. J. Grannis, and C. J. McDonald, “Privacy-preserving distributed queries for a clinical case research network”, in IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining, Maebshi City, Japan. Conferences in Research and Practice in Information Technilogy, Vol.14. C. Clifton and V. Estivill-Castro, Eds.

[7] B. Bloom. “Space/time trade-offs in hash coding with allowable errors”. Communications of the ACM, 13(7):422–426, Jul 1970.

[8] L. Fan, P. Cao, J. Almeida, and A. Broder. “Summary cache: a scalable wide-area Web cache sharing protocol”. IEEE/ACM Transactions on Networking, 8(3):281–293, Jun 2000.

[9] R. Agrawal, A. Evfimievski, and R. Srikant. “Information sharing across private databases”. In Proceedings of the 2003 ACM SIGMOD Int’l Conf. on Management of Data, San Diego, CA, 2003.
[10] L. Yaping, J.D. Tygar, and J. M. Hellerstein, “Private Matching”. Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.