A Way to More Efficient OT-Based Private Set Intersection?
Cuckoo Hash with More Buckets in the Bins

Huiyang He *ChengLin Li*
School of Cyber Science and Technology,
University of Science and Technology of China

1. Introduction & Problem Statement

1.1 Private Set Intersection

Private Set Intersection (PSI) is a technique that enables two party P_1, $holding\ set\ S_1$ and P_2, $holding\ set\ S_2$, to find the intersection of their sets, i.e. $S_1 \cap S_2$, without revealing any other information about their own set. PSI is usually considered as a specific area of *Secure Multi-party Computation* (MPC), which means there exists purposed methods that usually outperform the general circuit method and are more practical, but the general MPC method has better compatibility and thus can be combined with other general protocol to extend the functionality.

1.2 PSI & Privacy

It is obvious that PSI is closely related to big data privacy. Using PSI, not only do organizations retain their data property, but also the individuals get better protections of their confidential data. Two examples below to instantiate this viewpoint.

A common scene of applying PSI technic is measuring advertisement conversion rate. With the coming of the digital era, it is common for merchants to advertise on digital media. When the advertiser and the dealer calculate the fee, they must agree on an advertisement conversion rate. Usually, the first step to calculate advertisement conversion rate is to identify the people, who have seen the advertisement, have a transaction record in the merchant's database**.** This problem can be generalized as that the merchant holds a list of customers who have done a transaction and the advertiser holds a list of users who have seen the advertisement. Then they want to find the intersection of those two lists. This, of course, can be done by one of them sending its whole list to the other. However, in the real world, those data are valuable property and neither of them would be willing to provide the other with its own data. Thus, PSI solves the privacy problem by revealing only the intersection but no other information.

Another example of utilizing PSI to protect privacy is when doing the contact discovery. Contact discovery is a generic functionality for social media service nowadays. When a new user signs up for the service, the service provider might want to identify registered users who are in the contact of the new user. Nevertheless, neither is it possible to ask the service provider to present its complete database, nor should the user's privacy be undermined. This can also simply be accomplished by applying the PSI protocols. What is different is that the service provider now has a large set, but the user has a relatively small set, which means those two sets are unbalanced, and as a result, a protocol which aims at unbalanced set intersection might fit in better here.

Besides those, PSI is sometimes also a basic tool when one wants to apply other privacy preserving mechanics. For example, PSI can be used to do data alignment as a preparation for the privacy preserving machine learning.

1.3 An Overview of Oblivious Transfer-Based PSI

In [1], Pinkas etc. classified the OT protocols based on the technics. They overviewed the five classes, which are *naïve solution PSI*, *Public-Key-Based PSI*, *Circuit-Based PSI*, *Oblivious Transfer (OT)*

-*Based PSI* and *Third Party-Based PSI*. In this section, only an overview of the OT-Based PSI will be given.

Compared to Public-Key-Based PSI and Circuit-Based PSI, OT-Based PSI was suggested in a relatively late time. However, thanks to the IKNP OT extension [2] and KK OT extension [3] , OT-based PSI protocols can be relatively only a small factor slower than the insecure naïve hashing approach, and thus is more practical than other methods in some scenes.

In [4], Dong etc. proposed an OT-based PSI protocol by utilizing *Blossom Filter* (BF). They showed a new data structure called *Garbled Blossom Filter* (GBF) which is a combination of *Secret Sharing* (SS) and blossom filter. In this protocol, OT was used to merge the GBF hold by one party and the BF hold by the other. After obtaining the merged GBF, both of the parties can use the shares in it to check whether an element is in the intersection. It was shown that their protocol can be very efficient for large data sets.

In 2014, Pinkas etc. constructed a fancy OT-based PSI protocol based on OT extension [2] [3]. They start the construction with *private equality test* (PEQT), which consumes serval rows in the OT extension matrix. Then, the basic PEQT protocol was extended to *private set inclusion* protocol, which just applies the PEQT protocol multiple times. Finally, a prototype of their OT-Based PSI Protocol was shown: to find the intersection, one party just simply invokes the privates set inclusion protocol against the other party's set for every element in his own set. Using this prototype, the number of PEQTs needed is proportional to $O(n^2)$, where n is the size of the set. Recall that each PEQT consumes several rows of the OT extension matrix. Therefore, the less PEQT the protocol uses, the better performance it can achieve. To further reduce the PEQTs needed, they make use of hash table. Given that if two elements are equal, they must be hashed to the same bin using the same hash function. Consequently, the comparisons need only to be done between bins, each of which has the same index among the bins, instead of between the full sets. *Two-way cuckoo hash* was first chosen to minish the number of PEQTs in private set inclusion protocol. In more detail, it must be that one party uses the cuckoo hash, and the other uses simple hash to avoid leaking of side information. Nonetheless, later in [4] , *three-way cuckoo hash* has been proven to achieve better occupancy of the hash table on balanced sets, and thus less bins and comparisons are needed.

Then in 2017, [6] further boosted this OT extension-based protocol. They continued the thinking of [1] [4] and presented their efficient construction of *Oblivious Pseudorandom Function* (OPRF) using OT extension. OPRF is another atomic operation of MPC. In OPRF, the sender holds a seed s for the PRF, and the receiver has an input i for the PRF. For the receiver, it can only evaluate $f(s, i)$ and other $f(s, j)$ such that $i \neq j$ looks totally random to it (this kind of OPRF is also referred as Relaxed-OPRF, where the Strong-OPRF requires that the receiver cannot get any information about the send's seed). For the sender, it cannot infer any information about the receiver's input i, but since it has the seed, any value can be evaluated at the server side using the PRF f. In [6], each comparison using OPRF only consumes one row in the OT extension matrix, and therefore make the number of OTs irrelevant to the element size. This, again, make the PSI protocol has a performance closer to the unsecure naïve hash method and therefore more practical.

2 Research Question

2.1 Replacing the plain cuckoo hash in the protocol with cuckoo hash that hash more buckets.

In the state-of-art OT based PSI protocol, Pinkas etc. [1] [4] [5] utilize cuckoo hash to

reduce the number of comparisons, in other words, OPRFs needed. In more detail, they used two-way cuckoo hash, i.e. using two hash function to spread the elements, with 2.4n bins in [1], where n is the size of the set, but in [4], three-way cuckoo hash, i.e. using three hash functions to spread the elements, with 1.2n bins was showed to outperformance the two-way hash. A brief explanation of this is that three-way cuckoo hash has a higher occupancy of the hash table, so less bins are needed and therefore also less comparisons. However, it is also demonstrated by Fan etc. [6] that the occupancy of the hash table would also raise if the bins can hold more elements, and they call it cuckoo hash with buckets, where one buckets can hold one element and a bin can contains any more than one bucket, which means that the hash table is extended from a plain table to a table of higher dimension. As a result, cuckoo hash with more buckets can also be used to minish the bins need and therefore is also a good point to enhance the performance.

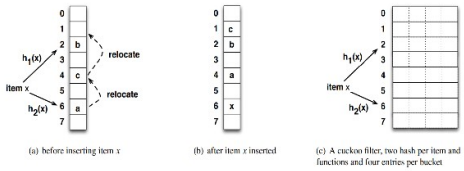

Figure 1. Illustration of cuckoo hashing

2.2 Utilizing the trick of quick two-way hashing in [6]

Fan etc. designed a new hashing scheme to calculate the index of bins the element should go to:

$$h_1(x) = hash(x)$$
$$h_2(x) = h_1(x) \oplus hash(x)$$

Upon inserting a new element into a cuckoo hash table, the old element in the bin will be kicked out and another insertion will be tried using the other hash function until an empty bin is found or the program reach a limit for the tires. Intuitively, the reinsert would happens a lot. Thus, using this new hashing scheme, the reinsert step of cuckoo hash can be speed up. This optimization can also apply to the OT-Based PSI protocol if we are using two-way cuckoo hash.

Table1. Overall communication for a larger number of hash functions h.

h	Util. [%]	#OTs	#Masks	Comm. [MB]	
				$n_1 = n_2$	$n_2 \ll n_1$
2	50.0	$2.00n_2$ t	$\mathbf{2}n_1\ell$	148.0	**17.0**
3	91.8	$1.09n_2$ t	$3n_1\ell$	**99.8**	25.5
4	97.7	$1.02n_2$ t	$4n_1\ell$	105.3	34.0
5	**99.2**	$\mathbf{1.01}n_2$ **t**	$5n_1\ell$	114.6	42.5

Communication is given for a) $n_1 = n_2 = 2^{20}$ and b)$n_2 = 2^8 << n_1 = 2^{20}$ elements of σ=32bit length. Utilization according to [6]

3 Methodology

3.1 Management

Our project timeline is basically consistent with or scheduled ahead of time with the course timeline. The critical path for our project is composed of four successive parts. Writing this proposal is the first part, and it may have been done when you read this. The second one is the realization of project content. The next one is experiment and comparison with published papers. Preparing for presentation will be the last part of them. The survey writing and other adjustments are not in the critical path as they can be done during the second to the fourth part.

As for team communication, considering that we are a small group of only two members, it is very easy for us to communicate. Meetings at least once a week can be held in classroom, café, or any suitable place. When necessary, additional meetings can be arranged at any time. We plan to communicate through WeChat or QQ as complements to the meetings.

Version Control is an especially important aspect for a successful project. We have created a repository on GitHub to hold papers, data, project code and any other needed materials. Consequently, git will be our version control tool.

3.2 Data Source & Evaluation Plan

(Notice: this section contains serval specific parameters of the experiment, which we might tends to change during the program)

The data used for test will be generated randomly using Cryptographic Pseudorandom Function. Data set size for both parties will be set to $n \in \{2^8, 2^{12}, 2^{16}, 2^{20}, 2^{24}\}$. We might use $\sigma = 32$ as the bit length for each element. For the security parameter, we do not have a comprehensive understand of them yet, so we decided to be consistent with [1] : a statistical security parameter $\lambda = 40$ and a symmetric security parameter $\kappa \in \{80, 128\}$.

All our experiments will be conducted on a server with Intel I9-10900K and 32G RAM. We will run both server and client on the same machine. The experiments will only simulate computation via LAN, which should be rational since we are only going to compare with the original protocol.

The performance will be evaluated using two metrics: computation cost (in seconds) and communication cost (in MB). The comparison of those two metrics should be straight-forward.

3.3 Challenges

Due to the lack of knowledge and limited time of research, we might face following challenges during our program:

- It is still not clear whether using the variant form of cuckoo hash would induce extra calculations to retain privacy or even worse, introduce new security problems that we cannot solve.
- Since the state-of-art OT based PSI protocol is already very efficient and practical, we might find that there is truly little performance boost using the variant of cuckoo hash.
- We did not find any clue of how to generate the test data in those papers about OT based PSI protocol and even worse, we noticed that their experiment platforms are high-end machines (32 cores CPU with 256G RAM). Thus, we need to figure out a way to assess our idea properly.
- The difficulty of replacing the plain cuckoo hash with bucket cuckoo hash is unknown to us, considering that we do not have time to even take a glance at the implementation yet.
- Before we can really understand the OT based PSI, we must learn the IKNP OT Extension and KK Extension, which are quite complicated.
- Since we are halfway pass this semester, there may not be enough time left for us to implement and evaluate both proposals

4 Impacts

This would be a very challenging word for us, considering that we only have step into this area for only about four weeks. We are aware that our idea of changing the protocol might even have a worse performance compared with the original one. But to our knowledge, we have not seen any other research that tries to use a variant form of cuckoo Hash in PSI protocol. We hope that we will have better sense of the way of using cuckoo hash in this specific protocol, and a more comprehensive understanding of PSI.

Citations

[1] B. Pinkas, T. Schneider and M. Zohner, "Faster Private Set Intersection Based on OT Extension," 2014.

[2] Y. Ishai, J. Kilian, K
. Nissim and E. Petrank, "Extending Oblivious Transfers Efficiently," 2003.

[3] V. Kolesnikov and R. Kumaresan, "Improved OT Extension for Transferring Short Secrets," 2013.

[4] B. Pinkas, T. Schneider, G. Segev and M. Zohner, "Phasing: Private Set Intersection using

Permutation-based Hashin," 2015.

[5] V. Kolesnikov, R. Kumaresan, M. Rosulek and N. Trieu, "Efficient Batched Oblivious PRF with Applications to Private Set Intersection," 2016.

[6] Kaminsky, B. Fan and D. G. A. Michael, "Cuckoo Filter: Practically Better Than Bloom," 2014.

[7] Changyu Dong, Zikai Chen and Liqun Wen, "When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol," 2013.