

Efficient Semi-Honest Multiparty Private Set-Intersection via Bloom Filters*

WORKING DRAFT: PLEASE DO NOT DISTRIBUTE

Roi Inbar and Eran Omri

Department of Computer Science, Ariel University
roikeman@gmail.com, omrier@ariel.ac.il

Abstract. [Eran's Note: TO DO]

1 Introduction

1. PSI as an important special case of MPC.
2. 2party PSI
3. A Bloom filter is a compact data structure to encode set of elements, with only a small false positive probability.
4. Efficiency and privacy.
5. Our Results
6. The full semi-honest model vs. the augmented semi-honest model. We obtain both, and the while total complexity does suffer from the distinction, there is no penalty when considering the maximum complexity of a single party. Our transformation is applicable to previous protocols as well.
7. The trusted curator model. An OT-free protocol.
8. Consider the following improvements in terms of complexity and of privacy.
 - (a) Sending seeds to a PRG instead of SSS shares.
 - (b)
9. – Japanese guys – did not implement. Bloom filter (not garbled). They use homomorphic encryption to compute the i 'th bit in the filter as the AND of all bits.

For example, PSI can be used by a governmental agency that performs a search over every flight passenger's list of every airline to assure none of the passengers is on its no-fly list. While both the governmental agency and airline are interested to keep their lists as private as possible it is not mandatory as it is a national security issue. For commercial companies that would like to intersect their lists of clients to increase their profits the demand for privacy becomes a necessity.

questions to Aner:

1. Should we come up with public keys.

*Research supported by ISF grant 544/13.

2. How to implement the OT extension.
3. How to implement a PRG

There has also been substantial work aiming at constructing concretely efficient secure protocols for specific functionalities. The motivation is to come up with protocols that are far more efficient than an application of the generic protocols for that functionality.

A problem of great importance and usability in the realm secure computation is that of *private set intersection* (PSI), motivated, e.g., by database operations. Here a set of parties, each holding a large private data-set, wish to compute the intersection over all data-sets. Much research was dedicated to the construction of PSI-tailored highly efficient protocols for the case of two-party set intersection. A survey of the abundance of works on efficient two-party PSI protocols is given in [?], including a classification of the underlying techniques. To our discussion, most relevant are the public-key-based PSI protocols (see, e.g., [? ? ?]) and the oblivious-transfer based and oblivious-pseudo-random-function based PSI protocols (see, e.g., [? ? ?]). Interestingly, in spite of the clear incentive, we are not aware of any published work on purpose-tailored protocols for multiparty PSI.

Construct concretely efficient protocols for multiparty private set intersection tolerating semi-honest/malicious adversaries.

Very relevant known results.

1.1 Our Contribution

1.2 More Related Work

2 Preliminaries

2.1 Notation

We use calligraphic letters to denote sets, uppercase for random variables, and lowercase for values. For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. Given a random variable (or a distribution) X , we write $x \leftarrow X$ to indicate that x is selected according to X . The support of a distribution D over a finite set S , denoted $\text{Supp}(D)$, is defined as $\{s \in S \mid D(s) > 0\}$. For a random variable X and a natural number n we let $X^n = (X^{(1)}, X^{(2)}, \dots, X^{(n)})$, where the $X^{(i)}$'s are i.i.d. copies of X .

3 Secure Multiparty Computation and the Private Set Intersection Functionality

3.1 Cryptographic Tools

3.1.1 Bloom Filters is a compact data structure for probabilistic set membership testing. A Bloom filter is a bits array in length of $m \in \mathbb{N}$. The operations

of inserting new item or checking if item is already inserted require predefined k uniform hash function set H , when $\forall h \in H \rightarrow h(item) = [0..m]$. Inserting given item is just simply as set to 1 all bits in the k -indexes that returned from the Hash functions set H . Checking if given item have inserted to the Bloom filter is actually checking if all bits in the k -indexes of this item is equal 1. The probability for false-negative is 0. The probability for false-positive is exist [?] and can be reduce to Negligible probability by increasing the array size. Bloom filter had great feature for intersection between two Bloom filters: given two $A[], B[]$ a m -sized arrays of Bloom filter that created by the same hash function set H . we can create the $C[]$ m array of bloom filter of $A \cap B$ by bit-wise AND operator.

Algorithm 1 (H) Bloom filter

```

1: procedure BFINSERT( $item, BFarray[], m, k, H$ )
2:   for  $i := 0..k$  do
3:      $hash \leftarrow H[i]$ 
4:      $index \leftarrow hash(item)$   $\triangleright$  index will be between 0 to  $m$ 
5:      $BFarray[index] \leftarrow 1$ 
6:   end for
7: end procedure
8: function BFCHECKING( $item, BFarray[], m, k, H$ )
9:   for  $i := 0..k$  do
10:     $hash \leftarrow H[i]$ 
11:     $index \leftarrow hash(item)$   $\triangleright$  index will be between 0 to  $m$ 
12:    if  $BFarray[index] = 0$  then
13:      return False
14:    end if
15:   end for
16:   return True
17: end function
18: function BFINTERSEC( $BFarray_1[], BFarray_2[], m$ )
19:   for  $i := 0..m$  do
20:      $BFintersection[i] \leftarrow BFarray_1[i] \wedge BFarray_2[i]$ 
21:   end for
22:   return  $BFarrayOfIntersec[]$ 
23: end function

```

3.1.2 Garbled Bloom Filters (GBF in shortly) is a new variant of Bloom Filters Data structure [?]. By expend any bit in the Bloom filter Array to String, we give up the compact attribute, but receive new feathers we will describe in the end of the section.

GBF is a array of $m \in \mathbb{N}$ fixed $\lambda \in \mathbb{N}$ length strings. The operations of inserting new item or checking if item is already inserted will require predefined k hash function set H as describe in 3.1.1.

Insert Given Item: for any index from the k -indexes we choose empty one and

keep it as *finalIndex*. for each one of other $k - 1$ indexes, if it empty we will fill it with λ length random string. Otherwise we will do noting. Then we fill the *finalIndex* with the bit-wise xor of all the $k - 1$ strings we fill already.

3.1.3 Threshold Secret Sharing Schemes

3.1.4 Garbled Bloom Filters with Shares By Using **simple** xor sharing of the GBF strings array we can use the same mechanism of GBF intersection and in the same time keeping the final intersection GBF unreadable unless all the shares will combined together.

3.1.5 Oblivious Transfer

4 Multiparty PSI Protocol with Augmented Semi-Honest Security

Given $d \in \mathbb{N}$ we define domain set of words as $\mathcal{D} = \{w | w \in \{0, 1\}^d\}$.

Let p_i be an entity who holds private subset $\mathcal{DB}_i \subset \mathcal{D}$. Just to simplify and **Without** limiting the generality we assume that all $|\mathcal{DB}_i| = n \in \mathbb{N}$, number of item in any of the private sets.

Given $\mathcal{P} = \{p_0, p_1, p_2, \dots, p_{|\mathcal{P}|}\}$ set of players who **want** calculate the set intersection $\mathcal{IS} = \mathcal{DB}_0 \cap \mathcal{DB}_1 \cap \mathcal{DB}_2 \cap \dots \cap \mathcal{DB}_{|\mathcal{P}|}$

p_0 will be the **leader** of the protocol. It have few additional work to do in the initialization and the end of the protocol. In the other parts of the protocol it will actually act like all the other **Participants**.

Initialization Step: p_0 will **Randomly** choose k hash function set H , and send it to any $p_i \in \mathcal{P} \setminus p_0$.

Any $p_i \in \mathcal{P}$ will:

- Create a local $GBF_{\mathcal{DB}_i} = \text{BuildGBF}(\mathcal{DB}_i, H, k, m, \lambda)$ using the H set (require for GBF intersection).
- Create a local $BF_{\mathcal{DB}_i} = \text{BuildBF}(\mathcal{DB}_i, H, k, m)$.
- Create a $|\mathcal{P}|$ GBF Shares i.e $GBF_{\mathcal{DB}_i}^0 \oplus GBF_{\mathcal{DB}_i}^1 \oplus \dots \oplus GBF_{\mathcal{DB}_i}^{|\mathcal{P}|} = \mathcal{DB}_i$.¹

OT Step:

any $p_i \in \mathcal{P}$:

- For each $p_j \in \mathcal{P} \setminus p_i$

¹It should be noted that every $GBF_{\mathcal{DB}_i}^o$ is equivalent to random String in length of $\lambda * m$.

- p_i act as OT sender and send to p_j m pairs λ strings $(x_{(i,j,b,0)}, x_{(i,j,b,1)})$ where $x_{(i,j,b,0)}$ is a uniformly λ -bit string and $x_{(i,j,b,1)} = GBF_{\mathcal{DB}_i}^j[b]$ where $b = 0 \dots m$
- p_i act as OT receiver while using its local BF_i as selection string. i.e for $b = 0 \dots m$, p_i will locally create $GBF_{\mathcal{DB}_j}^{*i}$ where $GBF_{\mathcal{DB}_j}^{*i}[b] = x_{(j,i,b,BF_i[b])}$

Creating Final Share Step:

Any $p_i \in \mathcal{P}$:

- Create GBF Share $GBF_{\mathcal{DB}_{intersec}}^{*i} = GBF_{\mathcal{DB}_0}^{*i} \oplus GBF_{\mathcal{DB}_1}^{*i} \oplus \dots \oplus GBF_{\mathcal{DB}_{|\mathcal{P}|}}^{*i}$.
- Send $GBF_{\mathcal{DB}_{intersec}}^{*i}$ to p_0 .

p_0 will learn the intersection:

- Create $GBF_{intersec} = GBF_{\mathcal{DB}_{intersec}}^{*0} \oplus GBF_{\mathcal{DB}_{intersec}}^{*1} \oplus \dots \oplus GBF_{\mathcal{DB}_{intersec}}^{*|\mathcal{P}|}$.
- For any $item \in \mathcal{DB}_0$ that $GBFQuery(GBF_{intersec}, item) = 1$ than $item \in \mathcal{IS}$

4.1 Security Analysis

Security proof of the augmented semi-hones protocol:

Theorem 2. *Given \mathcal{P} set of $|\mathcal{P}|$ parties, Let \mathcal{DB}_i be private set of n items from a predefined universe, $f_{\cap}()$ be the intersection function.*

Proof. If the OT_n^λ is secure the the simulator for the sender and receiver a exist. we can use them as subroutines when construction our simulators.

p_i 's **view**: we construct a simulator Sim_{p_i} that receives the p_i private input and generates the view of the p_i in the protocol. Given \mathcal{DB}_i , the simulator Sim_{p_i} uniformly chooses its random coins r and generate $BF_{\mathcal{DB}_i}$, $GBF_{\mathcal{DB}_i}$ and it's $|\mathcal{P}|$ shares. Then Sim_{p_i} invokes the $(|\mathcal{P}|-1)$ simulations of the OT as sender Sim_{snd}^{OT} with $GBF_{\mathcal{DB}_i}$ and receiver Sim_{rcv}^{OT} with $BF_{\mathcal{DB}_i}$ that is guaranteed to exist. Then create the final share. Finally Sim_{p_i} return the simulated view $(\mathcal{DB}_i, \sim sndOT, \sim rcvOT, \{GBF_{\mathcal{DB}_0}^{*i} \dots GBF_{\mathcal{DB}_{|\mathcal{P}|}}^{*i}\}, GBF_{\mathcal{DB}_i}, BF_{\mathcal{DB}_i}, GBF_{\mathcal{DB}_{intersec}}^{*|\mathcal{P}|})$

5 Multiparty PSI Protocol with Augmented Semi-Honest Security

6 Experimental Results

Bibliography