PRIVATE SET INTERSECTION -- REQUIREMENT ANALYSIS

Our base reference in implementing the Private Set Intersection Protocol is this reference:
Private Set Intersection in the Internet Setting From Lightweight Oblivious PRF, by Mellisa Chase from Microsoft Research, and Peihan Miao from Visa Research.

With our scientific supervisor, we concluded that this exposure to the protocol is an appropriate choice because it was presented at the conference *Advances in Cryptology - CRYPTO 2020*, a prestigious session of talk in cryptography and information security domain. Also, this paper was published in 2020, so we can consider that this paperwork belongs to actual interest.

The link to the article is:
https://eprint.iacr.org/2020/729?fbclid=IwAR1Co-HI0YgoJk0ZCxHXOsk6K4PffelpMrDkxS8gbxIskpD3WrCcEa7G-mE

And to YouTube presentation is:

https://www.youtube.com/watch?v=YWnaShAsFL4

NUCLEUS ALGORITHM COMPONENT:
The algorithm is presented below (from the reference):

0. P_1 and P_2 agree on security parameters λ, σ, protocol parameters m, w, ℓ_1, ℓ_2, two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{\ell_1}$ and $H_2 : \{0,1\}^w \to \{0,1\}^{\ell_2}$, pseudorandom function $F : \{0,1\}^\lambda \times \{0,1\}^{\ell_1} \to [m]^w$.

1. **Precomputation**
   - P_1 samples a random string $s \xleftarrow{\$} \{0,1\}^w$.
   - P_2 does the following:
     (a) Initialize an $m \times w$ binary matrix D to all 1's. Denote its column vectors by $D_1, \ldots, D_w$. Then $D_1 = \cdots = D_w = 1^m$.
     (b) Sample a uniformly random PRF key $k \xleftarrow{\$} \{0,1\}^\lambda$.
     (c) For each $y \in Y$, compute $v = F_k(H_1(y))$. Set $D_i[v[i]] = 0$ for all $i \in [w]$.

2. **Oblivious Transfer**
   (a) P_2 randomly samples an $m \times w$ binary matrix $A \xleftarrow{\$} \{0,1\}^{m \times w}$. Compute matrix $B = A \oplus D$.
   (b) P_1 and P_2 run w oblivious transfers where P_2 is the sender with inputs $\{A_i, B_i\}_{i \in [w]}$ and P_1 is the receiver with inputs $s[1], \ldots, s[w]$. As a result P_1 obtains w number of m-bit strings as the column vectors of matrix C (with dimension $m \times w$).

3. **OPRF Evaluation**
   (a) P_2 sends the PRF key k to P_1.
   (b) For each $x \in X$, P_1 computes $v = F_k(H_1(x))$ and its OPRF value $\psi = H_2(C_1[v[1]] \| \ldots \| C_w[v[w]])$ and sends ψ to P_2.
   (c) Let Ψ be the set of OPRF values received from P_1. For each $y \in Y$, P_2 computes $v = F_k(H_1(y))$ and its OPRF value $\psi = H_2(A_1[v[1]] \| \ldots \| A_w[v[w]])$ and outputs y iff $\psi \in \Psi$.

Figure 3: Our private set intersection protocol.

1. P_1 and P_2 perform w random OTs with message length m, where P_1 is the receiver with inputs choice bits $s[1], \ldots, s[w]$. As a result, P_2 gets w pairs of random messages $\{r_i^{(0)}, r_i^{(1)}\}_{i \in [w]}$ and P_1 gets w messages $\{r_i\}_{i \in [w]}$ where $r_i = r_i^{(s[i])}$.
2. P_2 does the following:
   (a) Let $\{r_i^{(0)}\}_{i \in [w]}$ form the column vectors of the matrix A and compute the matrix $B = A \oplus D$.
   (b) Compute $\Delta_i = B_i \oplus r_i^{(1)}$ for all $i \in [w]$ and send to P_1.
3. P_1 computes the matrix C as follows: if $s[i] = 0$ then set $C_i = r_i$; otherwise set $C_i = r_i \oplus \Delta_i$.

Figure 4: Step 2 of our PSI protocol instantiated using random OT.

There are two main pieces we are going to focus on: the nucleus of the algorithm and the communication part. An approach of common sense is to define two entities:
P1 - Sender
P2 - Receiver
As spotted in item no. 3, the communication must be ensured bidirectional, so an appropriate way should be a socket communication, in a P2P way (server can behave as a client and vice-versa)

As we can observe, there are four stages of the algorithm:

1. **Establishment phase:** in this phase, we are going to negotiate parameters, hash algorithms, and the PRF algorithm. We must ensure that we can call a battery of different hash and PRF functions, in order to test them which one is more efficient (*trial and error* concept). Here, we are going to apply the Design Pattern *Template*, in order to design different services for algorithms using multiple hashes functions.

2. **Precomputation phase:** in this phase, these two entities are going to set the corresponding implied parameters: for the **Sender**, we initialize a bit vector **s.** The **Receiver** builds a binary matrix **D**, a PRF key **k**, and builds a map for each element from its set, a map containing a PRF applied to a hash of each element.

3. **Oblivious Transfer phase**: in this phase, we are applying for the transfer protocol. In the reference, there are proposed two approaches, as observed above. The authors spot the fact that the second approach is more optimal than the first one. In this stage, the **Receiver** behaves as a Sender and vice-versa, in order to exchange samples for building different structures used for algorithms.

4. **OPRF Evaluation phase:** in this phase, these two entities shall behave as normally, exchanging the key generated at stage 2, mapping the elements from the **Sender's** set by applying a PRF applied to a hash of each element, and after that computing the value **psi** for each element, as explained above, and set it. The **Receiver** will compute a **psi** value for each value from its set, and find the elements from the **Receiver's** set which **psi** values are equal.

The correctness and the security of the algorithm are proved and explained in the article.

DATA LOADING PROCEDURE

A simple solution shall be a Python Interface for Receiver (Client), an one for Sender(Server).

Interaction with server shall occur like this:

A. Load data (in csv / xls format)
B. Start a server at a designated IP and PORT (which shall be displayed)

Interaction with client shall occur like this:

A. Connect to a server , introducing address and port
B. Load data (in csv / xls format)
C. Set parameters
D. Set algorithms
E. Send these parameters to server
F. Set columns to be account in algorithm
G. Set special columns to discriminate (optionally)
H. Set Criteria (display common elements / count common elements / etc)
I. Start client execution
J. Display result

This is a normal interaction flow. Of course, there could appear some corner cases which we must treat.

A first scenario is that the remote part with designated IP and Port cannot be contacted (Connection refused). In this case, an error message should display (*Cannot connect to source with IP: …… and PORT: ……...*)

Some scenarios belong to file issues. In case of no file loaded, an error message should be displayed. Also, if the format is unappropriate, and if the file does not respect the standard format:

C1 …… C2 …… C3 …… Cn
Data1 Data2 Data3 Datan

then an error message should be displayed.

Also, the button should not be available if column has not been selected or the criteria.