Private Set Intersection for Privacy Preserving Comparisons

Harry Bairstow

December 2022

Contents

1 Introduction

Private Set Intersection (PSI) is a cryptographic method to compare 2 sets without sharing the contents of the set. There are several sub-types of PSI that I explored. During the process of this document I explain everything I have learnt as well as some interesting use-cases I thought of. To finish I also show some examples of PSI in use with real companies.

2 Why Private Set Intersection?

When I was in Lisbon earlier this year for EthLisbon 2022 I attended a CASA (Chain Agnostic Standards Alliance) in-person meetup where I spoke to other attendees about several topics including (but not limited to): CAIP-25, ReCaps, etc. After the meetup I stayed to speak to a few of the others there about other personal and work projects where PSI was brought up as a topic someone else was learning about. After that I decided I would also investagate the topic and see what there is to know and find out; as it turned out PSI was a large problem with many varying solution from different entities big and small. I thought this could be an interesting field to know more about both for my work and personal projects so decided I would research it more for this project.

3 Research

3.1 Blog Posts

When researching this project I first decided to look at a series of blog posts I found from Decentralised Thoughts, although these were not peer reviewed they did include links to several papers used while doing their own research. I read the 2 blog posts that they wrote and that gave me a solid understanding of the root problem and how people have gone about solving it. The blog posts that I read were: Private Set Intersection a Soft Introduction by Yanai and Private Set Intersection #2 by Yanai.

3.2 Scaling private set intersection to billion-element sets

This is a paper which is cited within the blog posts mentioned above, I found the paper to be rather interesting as it broke down the problems that Microsoft faced and came up with solutions to them. Below I breakdown my understanding of the paper which furthered my understanding of PSI before I moved onto describing the problem I think PSI attempts to solve as well as potential uses. Within the first page of this document the authors, from Microsoft, University of Calgary and IBM Research, explain the problem they are trying to explore. That is to "examine the feasibility of private set intersection (PSI) over massive datasets." (Kamara et al. 2014) which shows that they are exploring the topic from the point of view of a massive corporation or government. This paper describes the problem that PSI solves as "two parties want to learn the intersection of their sets without revealing to each other any information about their sets beyond the intersection" (Kamara et al. 2014) which is similar to the description given in the blog posts on Decentralised Thought. After explaining the problems they go on to describe the new protocols which they have designed in the server-aided PSI setting.

3.2.1 Preamble

The paper describes 4 protocols and throughout uses some notation which is described below, as in the paper. For the purposes of all protocols in this section: k shall be equal to a computationally secure parameter and s shall be a statistically secure parameter. The authors also define S^λ, where $\lambda \geq 0$ as:

$$S^\lambda = \{x||1, \cdots, x||\lambda : x \in S\}$$

and $(S^\lambda)^{-\lambda} = S$. If $F : U \to V$ is a function, the S-evaluation of F is the set $F(S) = \{F(s) : s \in S\}$. The inverse of F is denoted as F^{-1} where $F^{-1}(F(S)) = S$. If $\pi : [|S|] \to [|S|]$ is a permutation, then the set $\pi(S)$ is a set of permutation elements of S according to π (assuming a natural order of elements). i.e.

$$\pi(S) = \{x_{\pi(i)}\} : x_i \in S$$

For the following section we are going to denote the union of S_1 and S_2 as $S_1 + S_2$ and the difference as $S_1 - S_2$. (Kamara et al. 2014)

3.2.2 Server-aided PSI with Semi-honest Server

This protocol ensures security by having a limited amount of ways to affect the outcome. This method is not fully safe as there are a few ways for both the clients and server to change the result. I have listed them below with an explanation of why each is bad and any comments I could think of for potential solutions; there is a conclusion at the end of this section which explores how well thought out my potential solutions were. The issues I saw with this protocol were:

- The server sends back incorrect information. This would mean the server is malicious, because of how this protocol works (discussed below) the server doesn't know what the items are but it could return each client a different intersection, return nothing or return random values. This would mean that the result from the server in this protocol can only be trusted if you some-how trust the server e.g. you have control of the server and it's code. However, that can only - in practice - be true for one of the parties involved as if both had control of the server they are most likely the same entity and in which case wouldn't have the need for PSI. This may not be true in all cases and as such this could be a good method as it isn't very complex.

- Clients could use different secret keys for the PRP (explained below); this wouldn't disclose any information as it would just mutate their data set differently to all other clients and as such they wouldn't intersect with any of the other data sets which would mean an empty set would be returned for the intersection.
- If they PRP could be reversed (or brute-forced/intercepted) then the Server could "decrypt" each element in the sets from the clients and then respond to them with an invalid response as discussed in the first point. A potential solution to this would be using a one-way algorithm that all clients can replicate in a way which can be compared

The protocol being this is as follows, to start the parties jointly generate a secret k-bit key K for pseudo random permutation (PRP). The below steps - from the paper - can be followed as blueprint:

1. P_1 generates a random secret k-bit key K and sends it to P_i for $i \in [2, n]$;
2. each party P_i for $i \in [n]$ sends $T_i = \pi_i(F_K(S_i))$, where F_K is the random permutation and π_i is the random permutation, to the server;
3. the server computes $I = \bigcap_{i=1}^{n} T_i$ and returns the result, I, to each party;
4. each party P_i computes $F_K^{-1}(I)$ and then outputs it.

After reading this I wrote the following as pseudo code that explains the above in a a simpler form.

```
// Party 1
let key = generate_key()
send_key_to_other_parties(key)

// All parties
let key = recieve_key()
let new_set = mutate_set(set, key)
send_to_server(new_set)
let result = get_result()
print(result)
```

This could be implemented fully but still has the problems discussed above, after reading about this method I feel like it is not a final solution so moved on to the next protocol.

3.2.3 Server-aided PSI with Malicious Security

This protocol is designed to combat the first issue I discussed for the prior protocol. That is the ability for the server to mutate the results without the clients being able to validate the result's authenticity. The paper suggests a way this can be solved which is to: "require each party P_i to augment its set S_i with λ copies of each element in a new set S_i^{λ}. The parties then generate a secret k-bit key K for a PRP F using a coin tossing protocol and evaluate PRP on their augmented sets resulting in $F_k(S_i^{\lambda})$. Finally they permute their set with a random permutation for a final result of $T_i = \pi_i(F_k(S_i^{\lambda}))$." (Kamara et al. 2014) As before the server then computes the intersection of $I = \bigcap_{i=1}^{n} T_i$ before sending the result back to each party. "Each party checks that $F_K^{-1}(I)$ contains all λ copies of every element and aborts if this is not the case". This now allows the client to check if the server has tampered with the returned intersection; the only way for the server to tamper now would be to remove all λ copies of each element which would be hard as they are mutated with a random permutation. This now allows for PSI with a potentially un-honest server where all clients can validate that their result is correct.

3.2.4 Fair Server-aided PSI

As an expansion to the Server-aided PSI with Malicious Security, this allows for a server to validate the honesty of all involved parties when the need to hide the contents of the sets from the server is not necessary. This does raise other questions such as the trust of the server as this may still be an untrusted environment

and what data would be considered safe to share with a server e.g. medical records should always be kept private, however, there is an increased need in that scenario to verify the honesty of other parties. This paper doesn't expand on this protocol other than the fact that the server would need the key K and the set S_i to check against T_i so it can validate the result I and report any tampering to the relevant client(s).

3.2.5 Intersection Size-Hiding Server-aided PSI

In the case that you cannot share the size of the intersection with the server this is a protocol that allows for PSI without that information being divulged. This protocol is different to the others where the actual PSI is conducted on P_1 where the server only acts to help P_1. I decided not to read too much about this protocol as I felt it was less interesting that some of the others that I could go on to explore and can save this for future reading and learning. So I left this protocol there with a basic understanding of the concept behind Intersection Size-Hiding for Server-aided PSI.

4 Overview of Private Set Intersection

This section contains an overview of PSI following the research in the above section.

4.1 The Problem

The problem Private Set Intersection tries to solve was very well described as: "Secure multi-party computation protocols enable multiple distrusting data holders, to jointly compute on their collected input, while revealing nothing to any party other than the output." (Le et al. 2019) this aligns with what I learnt about from "Scaling private set intersection to billion-element sets" and the use-cases explored below.

5 Potential & Existing Use-cases of PSI

5.1 Apple's CSAM Detection System

Describe this system as a system: "to accurately identify and report iCloud users who store known Child Sexual Abuse Material (CSAM) in their iCloud Photos accounts. Apple servers flag accounts exceeding a threshold number of images that match a known database of CSAM image hashes so that Apple can provide relevant information to the National Center for Missing and Exploited Children (NCMEC)." (Apple 2021) Apple's CSAM system is a well-known use of PSI which is documented in a paper that "describes the constraints that drove the design of the Apple private set intersection (PSI) protocol (Bhowmick et al. 2021). After I found that Apple used PSI I decided to read both the technical overview and the paper on Apple's PSI protocol to compare it to the other protocols I had read about as well as how Apple used PSI in a production environment.

What is CSAM For the context of this section CSAM is any exploitative imagery of children, CSAM stands for Child Sexual Abuse Material and is illegal in most developed countries. As such apple doesn't want to be responsible for hosting it on their servers.

How does Apple's CSAM Detection System work? The system Apple are creating has a few requirements:

1. Must preserve the user's privacy
2. There is a threshold known to both the server and client at which point the server learns the real values
3. "the intersection cardinality is below the threshold, it is desirable that the server have some uncertainty as to the exact size of the intersection and its contents" (Bhowmick et al. 2021)

Apple's Protocol Apple chose to create their own implementation of a variant of PSI, private set intersection with associated data (PSI-AD) which they expand upon with an extension called "threshold private set intersection with associated data (tPSI-AD)." (Bhowmick et al. 2021). Apple needed to use this protocol to ensure all their above requirements were fully met, below I show my understanding of the protocol as a brief summery. For this summary I build up-on what is described in my explanation of Server-aided PSI with Malicious Security. Let H be a threshold known to both the client and server and define the client as P_1. P_1 generates a hash, using $NH_1(S_1)$, for a set of images, S_1, it wishes to check, it compares it to a list of known CSAM hashes on the device. This then creates "Safety Voucher" which can be sent to the server. The server then performs it's own PSI matching to get a set of vouchers which match known CSAM, if that exceeds the threshold then the server can reconsruct the encryption key to gain access to the images for a moderator to review before sending to the relevant agencies.

Updates In a statement to Wired on the 7^{th} December 2022 Apple said

> "We have further decided to not move forward with our previously proposed CSAM detection tool for iCloud Photos. Children can be protected without companies combing through personal data, and we will continue working with governments, child advocates, and other companies to help protect young people, preserve their right to privacy, and make the internet a safer place for children and for us all." - (Hay Newman 2022)

This means that Apple have terminated their CSAM detection program and they will no longer be using PSI for this detection. There is other uses of PSI at Apple for password breach detection however, I chose not to explore that as this was to me a more interesting use of the technology.

Why was this interesting? I found this to be an interesting use of PSI because it expanded on the existing protocol to ensure that the safety of children using Apple's hardware and software remains intact. The fact that apple chose to use PSI because of it's ability to remain secure proves that PSI is a protocol for Privacy Preserving Comparisons.

5.2 Private Contact Discovery

A potential use case for PSI is contact discovery. A way to securely compare contacts with other users so that you can only ever see your shared contacts. In 2014, Signal wrote a blog post that investigated possible solutions to the problem of wanting to find others who used signal while staying private and secure. At the time they couldn't find a solution. I think that PSI could have been an appropriate solution for this, where 2+ devices takes their contacts and uses the Server-aided PSI with Malicious Security protocol to compare the contacts. There are flaws with this solution as both devices would need to have uploaded their contacts and know which other device they are looking for. Most likely they are looking for the other numbers in their contacts but Signal wanted a solution where the Signal servers didn't know what numbers were in your contacts. Signal did end up finding a solution (SGX Enclaves) in 2017 which they document here in a blog post. This could potentially be a good topic for future exploration.

6 Conclusion

To conclude, PSI is a useful tool for entities big and small who need to securely compare lists of data between 2 known clients. However, after researching it there are issues that could be addressed to make PSI easier for developers and users. I think that an interesting follow-up project could be to explore the expansions to PSI and think about a potential protocol that builds up-on PSI to make the experience better and easier for all involved.

References

Apple (2021), 'Csam detection - technical summary'.
URL: *https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf*

Bhowmick, A., Boneh, D., Myers, S., Talwar, K. & Tarbe, K. (2021), 'The apple psi system'.
URL: *https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf*

Hay Newman, L. (2022), 'Apple kills its plan to scan your photos for csam. here's what's next'.
URL: *https://www.wired.com/story/apple-photo-scanning-csam-communication-safety-messages/*

Kamara, S., Mohassel, P., Raykova, M. & Sadeghian, S. (2014), Scaling private set intersection to billion-element sets, *in* 'International conference on financial cryptography and data security', Springer, pp. 195–215.
URL: *https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/sapsi2.pdf*

Le, P. H., Ranellucci, S. & Gordon, S. D. (2019), Two-party private set intersection with an untrusted third party, *in* 'Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security', pp. 2403–2420.
URL: *https://eprint.iacr.org/2019/1338.pdf*

Yanai, A. (2020*a*), 'Private set intersection #2'.
URL: *https://decentralizedthoughts.github.io/2020-07-26-private-set-intersection-2/*

Yanai, A. (2020*b*), 'Private set intersection a soft introduction'.
URL: *https://decentralizedthoughts.github.io/2020-03-29-private-set-intersection-a-soft-introduction/*