Private Set Intersection: Past, Present and Future

Ionita Andreea
Faculty of Computer Science, Alexandru Ioan Cuza University, Iasi, Romania

Keywords: Private Set Intersection, Bilinear Maps, Symmetric Encryption, Secret Sharing, Modular Inverse.

Abstract: Privacy has been more and more difficult to obtain since the development of the Internet. Private set intersection has been and still is a subject of great interest. In this paper we present the state of the art for PSI and propose four new directions for PSI protocols based on bilinear maps, secret sharing, modular inverse and symmetric encryption. Although our proposals are not the best in terms of efficiency, we believe that there are many optimizations to be done to achieve performance competitive with the best known protocols.

1 INTRODUCTION

With the development of the Internet, privacy has become increasingly difficult to obtain. A specific problem of privacy that is hard to obtain is the intersection of two sets. Making the intersection of two sets is of course a easy thing to do, but when privacy comes in, it becomes a lot harder. Lets say we have a social network and we want to keep friendships private for each user. However, if two users want to see their mutual friends and preserve the privacy of non-mutual friends, they may need a private set intersection protocol. Of course, there is the trivial solution where both A and B give their input to a trusted third party in order to compute the intersection set. However, having a trusted third party only happens in fairy-tales. In terms of secure multiparty computation, private set intersection is a technique that allows two parties holding sets to compare encrypted versions of these sets in order to compute the intersection. It is desired that any of the two participants does not learn more than it should. We further present some applications for PSI:

1. **Covid19 App.** Every user collects tracing data on their mobile phones. If the local health authorities diagnose a user positive with the coronavirus, the user can share their data and transfer it to a server. Any other user of the app can now learn if they have potentially been in contact with a positive tested user by asking that server with the updated data.
2. **Mobile Messaging Service.** Someone who uses an application "APP" for mobile messaging service wants to know who is using the same application from its contacts. Of course, his privacy is very important to him and does not want the company who owns "APP" to know his contacts.
3. **Paternity Tests Privacy.**

Outline. We give definitions of PSI types and cryptography and present the adversary model in Section 2. In Section 3 we classify PSI protocols in public-key cryptography based PSI, generic-secure computation based PSI and OT-based PSI. Also, we focus on some hybrid protocols that we present in detail. In Section 4 we propose four new PSI approaches based on modular inverse, bilinear maps, secret sharing and self encryption. In the last section, we conclude our study about PSI.

2 PRELIMINARIES

2.1 Adversary Model

Semi-honest Model. This type of adversary is also known as *honest-but-curious*. As the name implies, a semi-honest party tries to be honest, so follows the protocol properly but is curious so it keeps a record of all its intermediate computations.
Security in Semi-honest Model. A protocol is considered to be secure in a semi-honest setting if and only if the view obtained in the execution is the same as the view obtained in the ideal model. In the case of semi-honest adversaries, the ideal model consists of each party sending its input to the trusted party, and the third party computing the corresponding output pair and sending each output to the corresponding party.
Malicious Adversary in Ideal Model. In this type of setting we consider the followings scenarios: • *Input:* Each party receives its input u. The honest party always sends u to the TTP, but the malicious party may

Andreea, I.
Private Set Intersection: Past, Present and Future.
DOI: 10.5220/0010525806800685
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 680-685
ISBN: 978-989-758-524-1

send an arbitrary u' or abort. • *TTP's answer:* In case the TTP obtained an input pair, it replies the first party with the wanted answer. Otherwise, it send "NULL". In case the first party (who received the answer) is malicious, it may stop the TTP to send the answer to the second party. If so, TTP sends "NULL" to the second party. If the TTP is not stopped, it sends the answer to the second party.

Malicious Adversary in Real Model. The malicious party can have any strategy implementable by a probabilistic polynomial-time algorithm to get information that he should not get.

Security in Malicious Model. Loosely speaking, the definition asserts that a secure two-party protocol emulates the ideal model. This is formulated by saying that admissible adversaries in the ideal model are able to simulate the execution of a secure real-model protocol under any admissible adversaries.

2.2 Cryptographic Primitives

Oblivious Pseudorandom Function. In an oblivious pseudorandom function, information is concealed from two parties that are involved in a PRF. That is, if Alice gives the input for a PRF to Bob, and Bob computes a PRF and gives the output to Alice, Bob is not able to see either the input or the output, and Alice is not able to see the secret key Bob uses with the PRF.

Permutation-based Hashing. Permutation-based hashing is a technique that allows the hashed elements to be converted in shorter strings that can be stored in the hash table for reducing storage space and computation complexity.

Cuckoo Hashing. Pagh and Rodler $(PaghandRodler, 2004)$ proposed a dynamization of a static dictionary that uses two hash tables: T_1 and T_2, each consisting of r words and two hash functions h_1, h_2. Every key $x \in S$ is stored either in cell $h_1(x)$ of T_1 or in cell $h_2(x)$ of T_2, but never in both.

Paillier Cryptosystem. is a probabilistic asymmetric algorithm for public key cryptography that is based on the intractability of decisional composite residuosity assumption. The problem of computing n-th residue classes is believed to be computationally difficult. The Paillier cryptosystem supports:homomorphic addition of plaintexts and homomorphic multiplication of plaintexts.

2.3 Efficient Data Structures

Bloom Filter: is a compressed data structure, with the the use of which, a set of n elements can be compressed to an array A of m bits.

Oblivious Keyword Search (OKS): is a structure that allows a user to search and retrieve the data associated with a chosen keyword in an oblivious way.

3 STATE OF THE ART IN PSI

We choose to use the classification made by (Pinkas et al., 2014) for PSI protocols: 1. Public-key cryptography-based PSI; 2. Generic secure computation-based PSI; 3. Oblivious transfer-based PSI. For each classification, we point out the main ideas of the type of protocol and the related work.

3.1 Public-key Cryptography-based PSI

Diffie Helman. In (Meadows, 1986) it is proposed the first protocol for private matching with the use of Diffie Helman key exchange scheme. In the beginning the server chooses a secret a and the client chooses a secret b. The idea is quite simple: server encrypts its input $\{x_1, x_2, ..., x_n\}$ as $\{x_1^a, x_2^a, ..., x_n^a\}$ and client encrypts its input $\{y_1, y_2, ..., y_n\}$ as $\{y_1^b, y_2^b, ..., y_n^b\}$. They send their encryptions to each other and compute $\{(y_1^b)^a, (y_2^b)^a, ..., (y_n^b)^a\}$ respectively $\{(x_1^a)^b, (x_2^a)^b, ..., (x_n^a)^b\}$. Because both a and b are secret, it is hard to compute g^{ab} from knowing (g^a, g^b).

RSA. The protocols based on RSA make use of strong RSA assumption.

The work by Cristofaro and Tsudik (De Cristofaro and Tsudik, 2010) addressed the problem of private set intersection by means of RSA signature. They adapt Ogata and Kurosava's adaptive Oblivious Keyword Search (Ogata and Kurosawa, 2004) for the PSI scenario as it follows: " instead of encrypting a string of 0's the server reveals the key as the hash of the signature for all elements in her set". Despite the efficiency of the protocol it comes with some issues. One of them is the fact that the double running of the protocol reveals to the client the changes made in the server's set.

3.2 Generic Secure Computation-based PSI

There are generic protocols that solve general problems of secure two party computation. More specifically, these protocols allow two parties to securely evaluate any function that can be expressed as a Boolean circuit. A Boolean circuit is a mathematical model for combinational digital logic circuits. It

provides a model for many digital components used in computer engineering. The generic protocol that we want to focus on is Yao's garbled circuits protocol that is used to evaluate any discrete function that can be represented as a circuit. That means, in terms of PSI, that both the Server and the Client put together their inputs and evaluate the intersection function with no leaking beyond what is implied by the intersection output.

In 2012 there was the belief that solutions for PSI using generic approaches were impractical so Huang, Evans and Katz decided to explore the validity of that belief. In (Huang et al., 2012) they perform PSI using Yao's generic garbled circuit approach obtaining efficient protocols assuming the semi-honest setting. They consider three classes of protocols for PSI based on Yas's garbled circuit technique which takes any boolean circuit C and yields a secure protocol for computing C. The main idea of the third is that each set is sorted locally and then obliviously merged into a single sorted list. Then each adjacent pair is compared and if the elements are equal, one of them is kept, and if not, the pair is replaced by a random value. The resulting list must be shuffled for not leaking any information. They concluded that protocols based on generic secure computation *"can offer performance that is competitive with the best known custom protocols"*.

3.3 Oblivious Transfer-based PSI

Oblivious Transfer. protocol is a type of protocol in which a sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece has been transferred.

In 2004 Freedman, Nissim and Pinkas (Freedman et al., 2004) were the first to introduce the concept of private set intersection solved using protocols based on oblivious polynomial evaluation that acts as a client-server communication where only the client learns the output. They obtain $O(k)$ communication overhead and $O(k \ln \ln k)$ computation for lists of length k. The protocol works as follows: C uses interpolation to generate the polynomial $P(y) = \sum_{u=0}^{k_c} \alpha_u y^u$ of degree k_c with roots $\{x_1, x_2, ..., x_{k_c}\}$ (his input). C encrypts the coefficients and sends them to S. $\forall y \in Y, S$ computes $Enc(P(y))$ using the homomorphic properties to evaluate the polynomial. Then, he chooses a random r and computes $Enc(rP(y) + y)$. Finally, he sends all the k_s ciphertexts permuted back to the client. C decrypts all k_s values and outputs the set intersection.

Freedman et al. (Freedman et al., 2016) tried to hit it big again in 2016. They proposed two protocols based on the use of homomorphic encryption. They obtain linear communication and computation overhead using both Paillier and ElGamal encryption schemes. They implement the protocols and analyse with different constants.

In contrast to previous protocols that used secure polynomial evaluation, Hazay and Lindell (Hazay and Lindell, 2008) came with a different approach in dealing with secure set intersection problem. They proposed the first protocols based on secure pseudorandom function evaluation. Their work was continued and improved by Jarecki and Liu in (Jarecki and Liu, 2009). However, the input domain of the PRF is restricted to polynomial size.

Chase and Miao in (Chase and Miao, 2020) make a positive progress on finding a new PSI protocol that achieves better communication and computation trade-offs. Their protocol is based on oblivious transfer, hashing, symmetric-key and bitwise operations. Their protocol achieves security in the random oracle model when the B party (the sender who gets nothing for output) is malicious.

Their PSI protocol's most important part is a new multi-point oblivious pseudorandom function protocol that is based on oblivious transfer and relies on symmetric-key, bitwise operations and hashing. The idea of the protocol is simple: since they can achieve multi-point *OPRF* while the second party has multiple elements as input and his output consists on all the elements evaluated then the set intersection can be easily computed in a private manner. The first party evaluates the *PRF* on every element in his set and send all the *PRF* values to the second party and by comparison these *PRF* values, he figures out the intersection of the two sets.

Ruan and Mao (Ruan and Mao, 2020) propose a new approach to PSI protocol, transforming the problem of the intersection of sets into the problem of finding roots of polynomials by using point-value polynomial representation. Their protocol stands out because of the lack of using a cryptosystem.

In this article, the authors represent the sets as polynomials' point-value pairs as follows: each party denotes n elements $(s_1, ..., s_n)$ as a n-degree polynomial $p(x) = \prod_{i=1}^{n}(x - s_i)$. They agree on a list of d elements $\{x_1, ..., x_d\}$ and evaluate their polynomials on these numerical values, making point-value pairs $\{(x_1, p(x_1)), ..., x_d, p(x_d))\}$. Because $n \leq d + 1$, the point-value pairs can be seen as a representation of each polynomial. Next they blind their polynomials by using pseudorandom function $\{(x_1, p(x_1) + z_1), ..., (x_d, p(x_d) + z_d)\}$ and exchange the blinded point-value pairs. The polynomial can be found by interpolation and the intersection by computing the

roots of the polynomial.

In (Kolesnikov et al., 2016) there is presented the fastest protocol in the semi-honest environment. In short, in this protocol A can learn only one output of each PRF function F_i, while B can learn any output he wants. Next A hashes his elements using Cuckoo Hashing, so he has two possible hash functions to use but chooses only one. On the other hand, B sends both of the hash functions used on his elements to encompass both possibilities of A.

The protocol in (Pinkas et al., 2020) uses the same conceptual setup as the semi-honest protocol from (Kolesnikov et al., 2016). Instead of placing each item into one bin or the other, A will secret share that item between the two bins. An item $a = s_1 \oplus s_2$ is associated with PRF outputs $F_1(s_1)$ and $F_2(s_2)$ and if these PRF outputs are XOR-ed together it will get the result $F_{12}(a)$. B can compute the function F_{12} on any element he wants so he simply computes the function computed by the associated elements of his input on his elements and send them to A.

4 OUR PROPOSALS FOR PSI

In this chapter we are going to present some ideas that can be starting points for further PSI protocols. On our best knowledge, these ideas are not presented in the existing literature. In the following protocols, $A's$ input is $S = \{a_1, a_2, ..., a_v\}$ and $B's$ input is $C = \{b_1, b_2, ..., b_w\}$.

4.1 Bilinear Maps Approach

For this construction idea we consider the following: 1. G_1 and G_2 are two (multiplicative) cyclic groups of prime order p; 2. g_1 is a generator of G_1 and g_2 is a generator of G_2; 3. ψ is a computable isomorphism from G_1 to G_2, with $\psi(g_1) = g_2$; 4. e is a computable bilinear map $e : G_1 \times G_2 \to G_T$ as described below.

A bilinear map is a map $e : G_1 \times G_2 \to G_T$ with the following properties: 1. For all $u \in G_1$, $v \in G_2$ and $a, b \in Z$, $e(u^a, v^b) = e(u, v)^{ab}$. This property is called *bilinearity*. 2. $e(g_1, g_2) \neq 1$; This property is called $non-degenerate$. Further we will present the PSI protocol:

A generates a secret parameter s_1 and public parameters p_1, g^{s_1} and B generates a secret parameter s_2 and a public parameter p_2. B computes $g^{\frac{s_2 p_2}{b_j}}$ for every $j = 1, 2, ..., w$.

To make the intersection, B must find $g_T^{s_1 p_1 s_2 p_2}$ and verify if

$$e(g^{a_i s_1 p_1}, g^{\frac{s_2 p_2}{b_j}}) = g_T^{s_1 p_1 s_2 p_2}. \quad (1)$$

$g_T^{s_1 p_1 s_2 p_2}$ can be computed in the following way: $e(g^{s_1}, g^{p_1 s_2 p_2}) = g_T^{s_1 p_1 s_2 p_2}$.

Correctness. is assured by the bilinear maps properties. More exactly, the bilinearity property $e(g^a, g^b) = e(g, g)^{ab}$. We denote $e(g, g)$ as g_T. When B receives $\overrightarrow{X}$ from A, he computes $e(X_i, g^{\frac{s_2 p_2}{b_j}})$ and verifies if the answer is $g_T^{s_1 p_1 s_2 p_2}$. In a more detailed way he computes $e(X_i, g^{\frac{s_2 p_2}{b_j}}) = e(g^{a_i s_1 p_1}, g^{\frac{s_2 p_2}{b_j}}) = e(g, g)^{a_i s_1 p_1 \frac{s_2 p_2}{b_j}}$. If a_i and b_j are the same, so they are part of the intersection, b_j is simplified by a_i and results exactly $e(g, g)^{s_1 p_1 s_2 p_2}$ which is $g_T^{s_1 p_1 s_2 p_2}$.

Security. In terms of security, this protocol has an exclusion issue, meaning that an attacker can determine for sure if an element z is not in A's set by checking for $i \in \{1, ..., v\}$ if $e(X_i, g^{\frac{s_2 p_2}{z}})$ is $g_T^{s_1 p_1 s_2 p_2}$. This issue is dangerous especially when the set's domain is small.

Efficiency. In terms of communication, there is transferred a vector of v elements from A to B. In terms of computational complexity, in the worst case, there are made w verification for each of the v elements from the vector, leading to $O(v \cdot w)$. This idea is expensive because of the use of the not so light bilinear maps but if a faster and lighter bilinear map variant would be found, this proposal could materialize.

4.2 Secret Sharing Approach

Secret sharing refers to methods for distributing a secret among a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number of shares are combined together; individual shares are of no use on their own.

Take into consideration that Enc is a homomorphic encryption scheme.

A constructs a secret sec starting from the subsecrets $\{a_1, ..., a_v\}$ (his input) with threshold v. Then he encrypts his input with a homomorphic encryption scheme to allow further operations over cryptotext. A sends to B the encryptions of his input along with sec.

In this moment B has A's elements encrypted, sec beside his input. B encrypts his elements and tries to find the intersection. B remove one element from A's set at a time and replaces it with each of his element to see if he can find the same secret as sec. If he does, the element that he put for the replacement is part of the intersection.

This sketch has many issues. Firstly there must be found a secret sharing scheme that does not take into consideration the order of the elements. The known secret sharing schemes take into consideration the order of the subsecrets in founding the secret. Secondly,

B has to perform $v \times w$ encryption to find the intersection, which is far away from the best time achieved by speciality literature for PSI computation. However, this idea is an interesting starting point, thinking about how cloud computing is more and more popular every day.

Correctness. is ensured by the fact that the secret *sec* can be exclusively reconstituted by all v elements from A's set. Considering that the threshold is v, *sec* cannot be reconstituted with less than v or with other elements than A's elements. When B replaces successively one element and replaces it with each of his elements, he can recover the secret *sec* if and only if the element replaced is the same with the replacement, so it is part of the intersection.

Security. A security issue is that an adversary can check is one element is not part of the A's set by replacing successively each element with x. Also, the elements are send encrypted with a homomorphic encryption scheme and because of that an attacker can verify if an element that he encrypts with the same homomorphic encryption scheme is part of A's set.

Efficiency. In terms of communication complexity, there are send only the secret *sec* and a vector with the encryptions of A. To find the intersection, the computational complexity is $O(v \cdot w)$ because for each element from the A's set, B takes each of its elements and verifies if is in the intersection.

4.3 Modular Inverse Approach

A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus m. More formally,

$$ax \equiv 1 \mod m.$$

Our idea of PSI protocol that uses modular inverse in presented below: A generates randomly a and sends it to B. For every b_i, B finds x_i such that

$$ax_i \equiv 1 \mod b_i.$$

B sends $\overrightarrow{x}$ to A. Once received the vector $\overrightarrow{x}$, A finds the intersection by multiplying a with each x_i, $i \in \{1,...,w\}$ at a time and see if there exists a_j, $j \in \{1,...,v\}$ such that $ax_i \equiv 1 \mod a_j$. If exists, the element a_j is in the intersection.

Correctness. Finding a modular inverse does not give a unique answer if the modulo is not prime. Because the modulo is himself the secret, it is hard to force it to be prime, so the correctness is hard to obtain. However a solution would be for A to send a vector of elements to B to find the modular inverse of those.

Security. A security issue consists in the fact that the protocol can be attacked by a man in the middle. An observer can obtain precious information about B's set by knowing a and $\overrightarrow{x}$. The attacker can find that elements of B are divisible by $ax_i - 1$ and if B has prime elements, the malicious one can find exactly its elements. Also, the information found by the man in the middle can be found also by A. As well, on successive run of the protocol, the man in the middle and A can find B's secrets with high probability or at least a divisor of the secret. (By making gcd of $(ax_i, a'x'_j)$.)

Efficiency. The communication complexity is $O(w)$ because of the send of vector $\overrightarrow{x}$. In terms of computational complexity, B performs w findings of modular inverse and A makes $v \cdot w$ multiplications to find the intersection.

4.4 Symmetric Encryption Approach

A generates a random value r and v random values r_i and computes $c_i = Enc_{a_i+r}(a_i||r_i||len(r_i))$ for each $i \in \{1,...,v\}$ then sends the encryptions (vector $\overrightarrow{c}$) along with r to B.

B tries to decrypt each encryption received from A using the key $b_j + r$, for every $j \in \{1,...,w\}$. If $Dec_{b_j+r}(c_i)$ is $b_j||random||l$ where random is of length l for one $j \in \{1,2,..,w\}$ and $i \in \{1,2,...,v\}$ then b_j is an element from the intersection.

4.4.1 Security Proof

We cannot prevent the server to send false data.

Let C, S be two sets from a predefined universe, $f_\cap$ be the set intersection function defined as:

$$f_\cap(C,S) = (f_C(C,S), f_S(C,S)) = (C \cap S, \wedge).$$

The correctness of the algorithm refers to the fact that the intersection output is the correct one. Intuitively, the intersection correctness in the proposed protocol comes from the fact that we add at the end of each secret (elements from the set) a random number and the length of the random number. In this way, after decrypting with the correct key, it is very precise where the secret ends, by eliminating the random number and its length. So, for each element of A we can verify with maximum precision if it is also in B's set.

Theorem 1. If *Enc* is a symmetric encryption algorithm and *Dec* is it inverse, a symmetric decryption algorithm, then the protocol securely realizes $f_\cap(C,S)$ in the semihonest model.

Proof. To prove 1 we will give the construction of two simulators Sim_A and Sim_B to simulate the view of each of the two parties. We want to show that the simulator's view is indistinguishable from the real world's protocol's view. Indistinguishability is a

property that means an adversary is unable to distinguish pairs of ciphertexts based on the message they encrypt. □

Simulator of A.

1. Sim_A is given the input of A: $S = \{a_1, ..., a_v\}$. Because A does not get any input from the protocol, neither is Sim_A;
2. Sim_A generates random $r \leftarrow_{rand} \{0,1\}^*$;
3. For every $i \in \{1, ..., v\}$, Sim_A generate a random $r_i \leftarrow_{rand} \{0,1\}^*$;
4. Instead of encrypting the input he was given, he randomly generates c_i and sends the vector $\overrightarrow{c}$ to B.

Simulator of B.

1. Sim_B is given the input of B: $C = \{a_1, ..., a_v\}$ and its output formed by the intersection $f_\cap(C,S)$.
2. Sim_B generates random what he should have received from A: $c_i \leftarrow_{rand} \{0,1\}^*$ for every $i \in \{1, ..., v\}$.

This property is achieved by having a different random for each secret, so the cyphertext is always different. Two secrets can have the same cyphertext with the probability of random generator giving the same number.

Unlinkability is also achieved by randomness. Taking into consideration that every time there is a new random generated, two actions are hard to be linked.

Efficiency. The communication complexity is $O(v)$ because A sends to B the vector $\overrightarrow{c}$. For each element from $\overrightarrow{c}$, B tries to decrypt with each of its keys. This lead to computational complexity of $O(v \cdot w)$.

5 CONCLUSIONS

In this overview, we presented the main aspects of private set intersection along with existing protocols. In the end we propose four new directions in terms of PSI that are not yet efficient but can be starting points for future protocols. The most promising is the proposal that makes use of symmetric encryption and we wish to improve its general efficiency in future. For the approach that uses secret sharing, in our future work we wish to find a secret sharing scheme that does not takes into consideration the order of the elements.

In conclusion PSI is still an object of great interest in speciality literature because of its multiple usage and our wish is to develop a protocol good in practice for solving one of the biggest internet's problems: the privacy.

REFERENCES

Chase, M. and Miao, P. (2020). Private set intersection in the internet setting from lightweight oblivious prf. In *CRYPTO*, pages 34–63. Springer.

De Cristofaro, E. and Tsudik, G. (2010). Practical private set intersection protocols with linear complexity. In *FC*, pages 143–159. Springer.

Freedman, M. J., Hazay, C., Nissim, K., and Pinkas, B. (2016). Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155.

Freedman, M. J., Nissim, K., and Pinkas, B. (2004). Efficient private matching and set intersection. In *EuroCrypt*, pages 1–19. Springer.

Hazay, C. and Lindell, Y. (2008). Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, pages 155–175. Springer.

Huang, Y., Evans, D., and Katz, J. (2012). Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*.

Jarecki, S. and Liu, X. (2009). Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *TCC*, pages 577–594. Springer.

Kolesnikov, V., Kumaresan, R., Rosulek, M., and Trieu, N. (2016). Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829.

Meadows, C. (1986). A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134. IEEE.

Ogata, W. and Kurosawa, K. (2004). Oblivious keyword search. *Journal of complexity*, 20(2-3):356–371.

Pagh, R. and Rodler, F. F. (2004). Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144.

Pinkas, B., Rosulek, M., Trieu, N., and Yanai, A. (2020). Psi from paxos: fast, malicious private set intersection. In *EuroCrypt*, pages 739–767. Springer.

Pinkas, B., Schneider, T., and Zohner, M. (2014). Faster private set intersection based on {OT} extension. In {*USENIX*} *Security 14*, pages 797–812.

Ruan, O. and Mao, H. (2020). Efficient private set intersection using point-value polynomial representation. *Security and Communication Networks*, 2020.