Quantum Information Processing (2021) 20:41
https://doi.org/10.1007/s11128-021-02987-4

Feasible private set intersection in quantum domain

Sumit Kumar Debnath[1] · **Kunal Dey**[1] · **Nibedita Kundu**[2] · **Tanmay Choudhury**[1]

Received: 29 May 2020 / Accepted: 1 January 2021 / Published online: 19 January 2021

Abstract
In the context of secure multi-party computation, private set intersection (PSI) is an important cryptographic primitive for performing joint operations on datasets in a privacy preserving manner. In particular, it allows the participants to privately determine the intersection of their private datasets. Most of the existing PSI protocols are based on traditional classical cryptosystems, which are proven to be vulnerable in quantum domain. This makes the requirement of quantum computer resistant PSI. Applying quantum cryptography in the design of PSI is an ideal approach to address these issues. In this paper, we present a quantum PSI (QPSI) relying on the basic quantum mechanics principles, which are resistant against well-known quantum attacks. Quantum resources in our QPSI are considered as single photons and we require to perform only simple single-particle projective measurements. These features make our QPSI more feasible to implement with the present technology, compared to the existing QPSI protocols, which adopt multi-particle entangled states and complicated quantum operators. On a more positive note, in our QPSI, only one time quantum communication and quantum computation allows execution of set intersection functionality multiple number of times, provided the client's set size remains same, while the existing QPSI protocols do not achieve this property.

Keywords Private set intersection · Quantum communication · Quantum computation · Long-term security

✉ Sumit Kumar Debnath
sdebnath.math@nitjsr.ac.in

[1] Department of Mathematics, National Institute of Technology Jamshedpur, Jamshedpur 831014, India

[2] Department of Mathematics, The LNM Institute of Information Technology, Jaipur 302031, India

1 Introduction

We are in a era where sharing of electronic information increases rapidly by concerning the data privacy. In this context, an extensively useful cryptographic primitive is private set intersection (PSI). It enables two parties, holding input sets X and Y, to secretly determine $X \cap Y$ without disclosing any extra information about their input sets. It has numerous real-life applications ranging from online advertising to threat detection, privacy contact discovery, data mining, law enforcement, etc. For example, PSI can be applied in the use of RFID tags for grocery stores to find the matching of the shoppers' personal preferences with the set of each item's adequate profiles that are held by the back-end-server of the store.

The problem of PSI has been widely studied in classical domain [1–3,5,8,11,13–16,18,24–28]. However, security of most of these PSI protocols is relied on the computational complexity assumptions, which are vulnerable in quantum domain due to Shor's algorithm [23]. Moreover, existing classical PSI protocols suffer long-term security issue. Furthermore, the current classical cryptosystem is not only potential threat for present, but a more serious and realistic threat for the future. For example, sensitive data (such as government documents, electronic health records) must remain secure for decades or even centuries. However, communicating classical encrypted version of these data over public channel may lose long-term security. This is because for the time being eavesdroppers may store encrypted data received from communications and wait for efficient quantum computer or classical algorithm. Once they are developed, eavesdroppers may apply those to decrypt the stored encrypted data. This implies there may be a very limited lifespan of confidentiality of messages. It makes the necessity of suitably countermeasures.

Post-quantum cryptography (PQC), which is the development of novel classical cryptosystems, can be thought as one approach to this direction since it remains secure against quantum computer under the existing quantum or classical algorithms. However, this may likely to provide a partial and temporary solution to the problem as in future some quantum or classical algorithms may be discovered that can break the PQC.

In contrast, quantum cryptography (QC) offers long-term security against an adversary having unlimited computational power based on the law of quantum physics. Thus, employment of QC in the development of elegant new tools and techniques towards designing efficient scalable PSI and its variants is essential. In this paper, our work is focused on the design and analysis of quantum PSI (QPSI) protocol. Our work is motivated by the oblivious set member decision protocol II of [20] in quantum domain.

The first QPSI was introduced by Shi et al. [21]. Later, Cheng et al. [4] claimed that the QPSI of [21] does not preserve fairness because client's query can be manipulated by a dishonest server. In order to prevent this drawback, [21] introduced a passive third party (TP) who is assumed to be fully trusted as it has access to the client's set intersection result. However, in real life, it is impossible to get such a fully trusted third party. In the following, Maitra [17] proposed an QPSI by extending the oblivious set member decision protocol of [22].

The existing works [4,17,21] provide only theoretical approach to solve the QPSI problem as they require "multi-particle entangled states" as quantum resources and

needs some "complicated oracle operators" and measurement in higher dimensional Hilbert space. However, preparing these resources and implementing these oracle operators are difficult with the present technologies. In contrast, this paper presents a practical and feasible QPSI with single photons and simple single-particle projective measurements in order to securely evaluate the set intersection functionality.

In our QPSI, once the raw key is shared between client and server using quantum communication, they may execute set intersection functionality multiple number of times with different sets, provided client's set size remains same. In other words, only one time quantum communication and quantum computation allows execution of set intersection functionality multiple number of times. While, each execution of set intersection functionality in the existing QPSI protocols requires quantum communication and quantum computation. Thus, compared to [4,17,21], our QPSI reduces quantum communication and quantum computation for multiple executions.

2 Proposed quantum PSI

A high-level overview: Our scheme uses the quantum key distribution (QKD)-based asymmetric key distribution technique of [7] between two parties, where only a few bits of the key is learned by one party, while the other party learns the whole key. Let the client C and the server S have the private sets $X = \{c_1, \ldots, c_u\} \subseteq Z$ and $Y = \{s_1, \ldots, s_w\} \subseteq Z$ respectively, where $Z = \mathbb{Z}_{N+1} - \{0\}$ for some positive integer N and $c_i \neq c_j$ for all $i, j \in \{1, \ldots, u\}$. The QPSI is run between two parties– a client C with private set X and a server S with private set Y. It completes in the following three phases: (i) Raw Key Sharing Phase, (ii) Asymmetric Key Sharing Phase, and (iii) Set Intersection Phase. During Raw Key Sharing Phase, an N-bit raw key Key_2 is shared in such a way that S has the knowledge of full $Key_2 = (\overline{k}_1, \ldots, \overline{k}_N)$ and C has the knowledge of u bits $\overline{K}_C = (b_1, \ldots, b_u) = (\overline{k}_{p_1}, \ldots, \overline{k}_{p_u})$ of Key_2, while S does not get any information about the position p_i of b_i in Key_2. In the following, during Asymmetric Key Sharing Phase, an N-bit key $Key_3 = (\widehat{k}_1, \ldots, \widehat{k}_N)$ is generated by giving a permutation ψ to the positions of the bits of Key_2 in such a way that the position set $\{p_1, \ldots, p_u\}$ of $\overline{K}_C$ is permuted to $\{c_1, \ldots, c_u\}$. This yields Key_3 as the shared key which is completely known to S and only $\widehat{K}_C = (\widehat{k}_{\psi(p_1)}, \ldots, \widehat{k}_{\psi(p_u)})$ is known to C. Finally, during Set Intersection Phase, S creates a set $G = F \oplus Key_3 = (f_1 \oplus \widehat{k}_1, \ldots, f_N \oplus \widehat{k}_N) = (g_1, \ldots, g_N)$ and sends it to C, where $f_i = 1$ if $i \in Y$ and $f_i = 0$ if $i \notin Y$. The client C in turn computes $\{h_1, \ldots, h_u\}$ with $h_i = g_{\psi(p_i)} \oplus \widehat{k}_{\psi(p_i)} = f_{\psi(p_i)} \oplus \widehat{k}_{\psi(p_i)} \oplus \widehat{k}_{\psi(p_i)} = f_{\psi(p_i)}$ and outputs the collection ξ of all $\psi(p_i)$'s for which $h_i = 1$ as $X \cap Y$. We now describe the three phases in more detail.

Protocol 1 *QPSI*

Raw Key Sharing Phase

1. *In order to share an $N + \lambda$-bit key $Key_1 = \{k_1, \ldots, k_{N+\lambda}\}$, the server S and the client C involve in QKD based asymmetric key distribution technique of [7], which enables S to learn the whole key Key_1 and C to learn $u + \lambda$ bits*

K_C *of* Key_1. *Note that we need to set* $\theta = \sin^{-1}\left(\sqrt{\frac{2(u+\lambda)}{N+\lambda}}\right)$ *in [7] for this key distribution. Instead of [7], if we use QKD based asymmetric key distribution technique of [12] then we require to set* $\lambda = (N - 4u)/3$.

2. *The client C randomly selects* λ *bits from its part and asks S to announce the bits of the corresponding positions. The client C then compares the announced bits with its record to check S's honesty. If the announced bits are exactly same as those of C's part then they continue to the next step; otherwise, if S is found to be dishonest then C terminates the execution.*
3. *Both of C and S delete the compared bits from their recorded parts so that C remains with u bits and S remains with N bits.*
4. *The server S updates the positions of the remaining N bits after removing the positions of the deleted bits from* Key_1 *and sets the updated N bits as new key* $Key_2 = (\overline{k}_1, \ldots, \overline{k}_N)$. *For example,* $\lambda = 2$, *and bit 3* (k_3) *and bit 6* (k_6) *are deleted. Then* $\overline{k}_i = k_i$ *for* $i = 1, 2$, *bit 4* (k_4) *will be updated to bit 3, i.e.,* $\overline{k}_3 = k_4$, *bit 5* (k_5) *will be updated to bit 4, i.e.,* $\overline{k}_4 = k_5$ *and bit i* (k_i) *will be updated to bit* $i - 2$, *i.e.,* $\overline{k}_{i-2} = k_i$ *for* $i = 7, \ldots, N + 2$.
5. *On the other hand, C also updates the positions of its remaining bits using the similar technique as of S and sets the updated u bits by* $\overline{K}_C = (b_1, \ldots, b_u)$. *For example, if* $K_C = (k_1, k_3, k_4, k_6, k_{10}, k_{13})$, *and* k_3 *and* k_6 *are deleted as earlier then* $b_1 = \overline{k}_1$, $b_2 = \overline{k}_3 = k_4$, $b_3 = \overline{k}_8 = k_{10}$ *and* $b_4 = \overline{k}_{11} = k_{13}$. *Let us denote the position of* b_i *in new key* Key_2 *as* p_i, *i.e.,* $\overline{k}_{p_i} = b_i$ *for* $i = 1, \ldots, u$. *Therefore* $\overline{K}_C = (b_1, \ldots, b_u) = (\overline{k}_{p_1}, \ldots, \overline{k}_{p_u})$.

Asymmetric Key Sharing Phase

1. *The client C randomly chooses a permutation* ψ *over* $\{1, \ldots, N\}$ *so that the elements of* $\{p_1, \ldots, p_u\}$ *are mapped to the elements of* $\{c_1, \ldots, c_u\}$ *in a random order. It then applies the permutation* ψ *to the position set* $\{p_1, \ldots, p_u\}$ *of its bit-string* $\overline{K}_C = (b_1, \ldots, b_u) = (\overline{k}_{p_1}, \ldots, \overline{k}_{p_u})$ *to get the updated bit-string as* $\widehat{K}_C = (\widehat{k}_{\psi(p_1)}, \ldots, \widehat{k}_{\psi(p_u)})$ *and sends the permutation* ψ *to S, where* $\widehat{k}_{\psi(p_i)} = \overline{k}_{p_i}$ *for* $i = 1, \ldots, u$. *Note that the set* $\{c_1, \ldots, c_u\}$ *is same as the set* $\{\psi(p_1), \ldots, \psi(p_u)\}$ *in some order.*
2. *On receiving* ψ, *the server S applies* ψ *to the position set* $\{1, \ldots, N\}$ *of its key* Key_2 *to get the updated key as* $Key_3 = (\widehat{k}_1, \ldots, \widehat{k}_N)$, *where* $\widehat{k}_{\psi(i)} = \overline{k}_i$ *for all* $i = 1, \ldots, N$.

Set Intersection Phase

1. *The server creates a bit-string* $F = (f_1, \ldots, f_N)$ *of length N, where*

$$f_i = \begin{cases} 1, & \text{if } i \in Y \\ 0, & \text{if } i \notin Y. \end{cases}$$

It then computes $G = F \oplus Key_3 = (f_1 \oplus \widehat{k}_1, \ldots, f_N \oplus \widehat{k}_N) = (g_1, \ldots, g_N)$ *and sends G to C.*

2. *The client C, on receiving* $G = (g_1, \ldots, g_N)$*, chooses an empty set* ξ *and does the following for* $i = 1, \ldots, u$*,*
   (a) computes $h_i = g_{\psi(p_i)} \oplus \widehat{k}_{\psi(p_i)} = f_{\psi(p_i)} \oplus \widehat{k}_{\psi(p_i)} \oplus \widehat{k}_{\psi(p_i)} = f_{\psi(p_i)}$*,*
   (b) if $h_i = 1$ *then includes* $\psi(p_i)$*, i.e., an element* c_j *of X in* ξ *since the set* $\{c_1, \ldots, c_u\}$ *is same as the set* $\{\psi(p_1), \ldots, \psi(p_u)\}$ *in some order.*
   Finally, outputs ξ *as the intersection* $X \cap Y$*.*

3 Security analysis

The basic security requirements of a PSI protocol, run between a client C with input X and a server S with input Y, for the functionality $F_{PSI} : (X, Y) \rightarrow (X \cap Y, \perp)$ are

1. *Correctness* The client C should output the exact (possibly empty) intersection, i.e., $X \cap Y$, on completion of the protocol.
2. *Client's privacy* The client C should get only $X \cap Y$ at the end of the interaction, not more than that.
3. *Server's privacy* The server S should get nothing ($\perp$) at the end of the interaction.

We describe below each of these security properties for our QPSI.

Correctness In order to show the correctness of the proposed QPSI, it is required to prove that $\xi = \{\psi(p_i) | h_i = 1; i = 1, \ldots, u\}$ gives the intersection $X \cap Y$. Note that the set $X = \{c_1, \ldots, c_u\}$ is same as the set $\{\psi(p_1), \ldots, \psi(p_u)\}$ in some order, i.e., $X = \{\psi(p_1), \ldots, \psi(p_u)\}$. Moreover, for $i = 1, \ldots, u$,

$$
\begin{aligned}
& h_i = 1 \implies g_{\psi(p_i)} \oplus \widehat{k}_{\pi(p_i)} = 1 \\
& \implies f_{\pi(p_i)} \oplus \widehat{k}_{\pi(p_i)} \oplus \widehat{k}_{\psi(p_i)} = 1 \\
& \implies f_{\psi(p_i)} = 1 \implies \psi(p_i) \in Y \text{ using the construction of } f \\
& \implies \psi(p_i) \in X \cap Y
\end{aligned}
$$

Client C's privacy The classical message sent by C to S is only the permutation ψ. By looking into the permutation ψ, the server S cannot retrieve any information about $X = \{c_1, \ldots, c_u\}$ since it does not have knowledge about $\{p_1, \ldots, p_u\}$. We will prove that S can determine $\{p_1, \ldots, p_u\}$ at most with negligible probability $\frac{1}{2^\lambda}$ for security parameter λ. In order to determine $\{p_1, \ldots, p_u\}$, during the creation of asymmetric key, a dishonest server S may apply entangle-measure attack by preparing a state of two qubits $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, where C receives the first qubit and S keeps the second qubit in its register. In the following, to obtain some information related to the conclusiveness of C's measurement, S will measure the state in its register. However, following [7,12], one can say that these attacks bring bit errors. In other words, if S obtains information related to the conclusiveness of C's bits, then S will be unable to get the information about C's recorded bit values. That is, S cannot simultaneously obtain both the correct bit value and the address of the correct measurement basis of C. Thereby, S cannot simultaneously get the correctly measured bit value k_i of C and the associated address i. Note that in our QPSI, in order to verify the honest behavior of S, the client C compares λ

measurement bit values with the corresponding bits declared by S. Therefore, a dishonest S will pass the honest test atmost with negligible probability $\frac{1}{2^{\lambda}}$. Thus we may conclude that C's privacy is preserved.

Server S's privacy In order to get more information about Y than $X \cap Y$, the client C needs to store more bit values in K_C during asymmetric key creation. In order achieve this goal, C may store the qubits obtained from Bob and utilize more effective measurements on those qubits after Bob's declaration during the first step of Raw Key Storing phase [7,12].

We first consider C's simple measurement i.e, for each qubit received by C. For instance, if S declared 0 in order to announce that a qubit is in $\{|0\rangle, |0^{'}\rangle\}$ then C carries out the optimal unambiguous state discrimination (USD) measurement [10,19] for identifying the state of the qubit. The USD measurement's success probability is bounded by $1 - F(\rho_0, \rho_1)$ for the fidelity $F(\rho_0, \rho_1)$ between the two states to be discriminated. Following [7], we may conclude that the probability is $p_{USD} = 1 - \langle 0|0^{'}\rangle = 1 - \cos\theta$. Note that due to USD measurement, the advantage of C is negligible as compared to the advantage in projective measurement, where $p = \sin^2\theta/2$ is the probability (especially for small θ).

According to [12], C may also do joint measurement on the k qubits which contribute to the final key's element. In other words, without distinguishing the individual bit values of the raw key, C wishes to directly retrieve the bit value of the final key. There are two possible measurements which can be utilized by C for this condition. One of them is minimal error-probability measurement of Helstrom, i.e., the measurement which distinguishes two quantum states having highest information gain [6,9]. The probability of distinguishing two equally likely quantum states ρ_0, ρ_1 and guessing the correct state is bounded by $p_G = \frac{1}{2} + \frac{1}{2}D(\rho_0, \rho_1)$ for the trace distance $D(\rho_0, \rho_1)$ between ρ_0 and ρ_1. Thereby, the probability of perfectly determining a final key bit is at most $p_G = \frac{1}{2} + \frac{1}{2}sin^k\theta$ for C. Note that the probability is close to 1/2 when θ is small.

The other measurement that can be performed by C is USD measurement. In this case, it is possible to obtain the success probability of discriminating the two k-qubit mixed states associated to odd and even parity in a unambiguous way and this probability rapidly declines with k. According to [7], C's advantage by joint USD is distinctly decreased when θ is small.

Note that in our scheme $N \gg u$. As a consequence θ is very small. Thus, following [7,12], we can say that it is impossible for C to store more bit values than $u + \lambda$. Hence, we may conclude that the server S's privacy is preserved.

Necessity of key reordering in Asymmetric Key Sharing phase If the client C does not apply the permutation to reorder the raw key during Asymmetric Key Sharing phase, then C would be unable to determine the key bits for the positions $c_1, \ldots, c_u$. In other words, C would not get $\{\widehat{k}_{c_1}, \ldots, \widehat{k}_{c_u}\}$. Thereby, it would not be possible for C to determine the intersection. This is because to determine c_i, it is necessary to have the knowledge of $\widehat{k}_{c_i}$ for $i = 1, \ldots, u$. Thus, the key reordering in Asymmetric Key Sharing phase becomes necessary.

4 Efficiency analysis

In our QPSI, S and C may involve in multiple execution of set intersection functionality with the same raw key, provided the set size of the C is always u. Note that quantum communication and quantum computation in our QPSI are required only in the Raw Key Sharing Phase. Thus, only one time quantum communication and quantum computation is sufficient for multiple execution of set intersection functionality. We describe below the round complexity, communication complexity and computation complexity of our QPSI.

Round complexity The "Raw Key Sharing Phase" requires 5 rounds, "Asymmetric Key Sharing Phase" requires 1 round and "Set Intersection Phase" requires 1 round.
Communication complexity The server S needs to send $O(N+\lambda)$-qubits as quantum communication and $O(N+\lambda)$- classical bits during "Raw Key Sharing Phase", and N classical bits in "Set Intersection Phase". On the other side, C requires to send $O(N+\lambda)$ classical bits in "Raw Key Sharing Phase" and a permutation π during "Asymmetric Key Sharing Phase".
Computation complexity The client C needs to compute $O(N+\lambda)$ projective measurements in two-dimensional Hilbert space during "Raw Key Sharing Phase", one permutation ψ in "Asymmetric Key Sharing Phase" and u XOR operations in "Set Intersection Phase". On the other hand, S has to compute the permutation ψ during "Asymmetric Key Sharing Phase" and N XOR operations in "Set Intersection Phase".

We refer Table 1 for a comparative summary of our work with the existing QPSI protocols.

5 Conclusion

In this work, we proposed an unconditionally secure two-party set intersection protocol utilizing the QKD based asymmetric key distribution technique of [7] as its building blocks. Its security depends on the basic principles of quantum mechanics, which are resistant against well-known quantum attacks. Unlike the classical PSI protocols, our QPSI offers long-term security as it uses quantum cryptography. Compared to the existing QPSI protocols [4,17,21], our QPSI is more feasible to implement with the present technology as single photons are considered as quantum resources and simple single-particle projective measurements are required to perform. Furthermore, in contrast to [4,17,21], our QPSI allows multiple execution of set intersection functionality by doing only one time quantum communication and quantum computation, which is required during raw key distribution.

Table 1 Comparison of different QPSI protocols

Protocol	Ours	[21]	[4]	[17]
Quantum resource	SP	MPES	MPES	MPES
Complicated oracle operators	Not required	Required	Required	Required
Dimension of the Hilbert Space	2	N	N	N
Simple single-particle projective measurements	Yes	No	No	No
Multiple execution of set intersection functionality with only one time quantum communication and quantum computation	Yes	No	No	No
Intersection cardinality revealed to server	No	No	No	Yes
Communication	$O(N+\lambda)$-QB $O(N+\lambda)$-CB	$O(u \log N)$-QB	$O(u \log N)$-QB	$O(u+l) \log N$-QB $O(\alpha(\log l + u \log N))$-CB
Computation	$O(N+\lambda)$	$O(u)$	$O(u)$	$O(N+l)$
Round complexity in set intersection phase	1	2	3	4

single photons = SP, multi-particle entangled states = MPES, CB = classical bits, QB = quantum bits, l, λ = security parameter, N = the size of the whole set, u = client's set size, w= server's set size, α = cardinality of the intersection

A Toy example

Let $N = 22$, $\lambda = 2$, $X = \{c_1, c_2, c_3, c_4\} = \{3, 7, 10, 12\}$ and $Y = \{s_1, \ldots, s_8\} = \{1, 3, 4, 7, 9, 11, 14, 15\}$. Then $u = 4$, $v = 8$.

Raw Key Sharing Phase

1. The asymmetric key distribution technique of [7] can be applied by setting $\theta = \pi/4$ so that C can learn only $(N + \lambda)\sin^2\theta/2 = 24/4 = 6 = (u + \lambda)$ bits (namely K_C) of the whole key Key_1 of size $(N + \lambda) = 24$. Let $Key_1 = (k_1 = 1, k_2 = 1, k_3 = 0, k_4 = 0, k_5 = 1, k_6 = 1, k_7 = 0, k_8 = 0, k_9 = 1, k_{10} = 1, k_{11} = 0, k_{12} = 0, k_{13} = 1 k_{14} = 1, k_{15} = 0, k_{16} = 0, k_{17} = 1, k_{18} = 1, k_{19} = 0, k_{20} = 0, k_{21} = 1, k_{22} = 1, k_{23} = 0, k_{24} = 0)$ and $K_C = (k_1 = 1, k_3 = 0, k_4 = 0, k_6 = 1, k_{10} = 1, k_{13} = 1)$.
2. Suppose C selects third bit $k_3 = 0$ and sixth bit $k_6 = 1$, and asks S to announce the bits of the corresponding positions. S then announces $k_3 = 0$ and $k_6 = 1$. The client C then compares the announced bits with its record to check S's honesty.
3. Both of C and S delete the compared bits $k_3 = 0$ and $k_6 = 1$ from their recorded parts so that C remains with $u = 4$ bits $(k_1 = 1, k_4 = 0, k_{10} = 1, k_{13} = 1)$ and S remains with $N = 22$ bits $(k_1 = 1, k_2 = 1, k_4 = 0, k_5 = 1, k_7 = 0, k_8 = 0, k_9 = 1, k_{10} = 1, k_{11} = 0, k_{12} = 0, k_{13} = 1 k_{14} = 1, k_{15} = 0, k_{16} = 0, k_{17} = 1, k_{18} = 1, k_{19} = 0, k_{20} = 0, k_{21} = 1, k_{22} = 1, k_{23} = 0, k_{24} = 0)$.
4. Therefore, $Key_2 = (\overline{k}_1 = k_1 = 1, \overline{k}_2 = k_2 = 1, \overline{k}_3 = k_4 = 0, \overline{k}_4 = k_5 = 1, \overline{k}_5 = k_7 = 0, \overline{k}_6 = k_8 = 0, \overline{k}_7 = k_9 = 1, \overline{k}_8 = k_{10} = 1, \overline{k}_9 = k_{11} = 0, \overline{k}_{10} = k_{12} = 0, \overline{k}_{11} = k_{13} = 1, \overline{k}_{12} = k_{14} = 1, \overline{k}_{13} = k_{15} = 0, \overline{k}_{14} = k_{16} = 0, \overline{k}_{15} = k_{17} = 1, \overline{k}_{16} = k_{18} = 1, \overline{k}_{17} = k_{19} = 0, \overline{k}_{18} = k_{20} = 0, \overline{k}_{19} = k_{21} = 1, \overline{k}_{20} = k_{22} = 1, \overline{k}_{21} = k_{23} = 0, \overline{k}_{22} = k_{24} = 0)$, $\overline{K}_C = (b_1, b_2, b_3, b_4) = (\overline{k}_1 = k_1 = 1, \overline{k}_3 = k_4 = 0, \overline{k}_8 = k_{10} = 1, \overline{k}_{11} = k_{13} = 1)$ and $p_1 = 1$, $p_2 = 3$, $p_3 = 8$, $p_4 = 11$.

Asymmetric Key Sharing Phase

1. The client C randomly chooses a permutation ψ over $\{1, \ldots, 22\}$ so that the elements of $\{p_1, p_2, p_3, p_4\} = \{1, 3, 8, 11\}$ are mapped to the elements of $\{c_1, c_2, c_3, c_4\} = \{3, 7, 10, 12\}$ in a random order. Let $\psi(p_1) = c_2$, $\psi(p_2) = c_3$, $\psi(p_3) = c_1$, $\psi(p_4) = c_4$ and $\psi(i) = j$ for $i \in \{1, \ldots, 22\} \setminus \{1, 3, 8, 11\}$ and $j \in \{1, \ldots, 22\} \setminus \{3, 7, 10, 12\}$ in some order. Then $\psi(1) = 7$, $\psi(3) = 10$, $\psi(8) = 3$ and $\psi(11) = 12$. The client C applies the permutation ψ to the position set $\{p_1, p_2, p_3, p_4\} = \{1, 3, 8, 11\}$ of its bit-string $\overline{K}_C = (\overline{k}_1 = 1, \overline{k}_3 = 0, \overline{k}_8 = 1, \overline{k}_{11} = 1)$ to get the updated bit-string as $\widehat{K}_C = (\widehat{k}_{\psi(1)}, \widehat{k}_{\psi(3)}, \widehat{k}_{\psi(8)}, \widehat{k}_{\psi(11)}) = (\widehat{k}_7, \widehat{k}_{10}, \widehat{k}_3, \widehat{k}_{12}) = (\overline{k}_1, \overline{k}_3, \overline{k}_8, \overline{k}_{11}) = (1, 0, 1, 1)$. Therefore, $\widehat{k}_7 = 1, \widehat{k}_{10} = 0, \widehat{k}_3 = 1, \widehat{k}_{12} = 1$.
2. On receiving ψ, the server S applies ψ to the position set $\{1, \ldots, 22\}$ of its key Key_2 to get the updated key as $Key_3 = (\widehat{k}_1, \ldots, \widehat{k}_{22})$, where $\widehat{k}_j = \overline{k}_i$ if $\psi(i) = j$ for all $i, j \in \{1, \ldots, 22\}$. Therefore, in Key_3, we have $\widehat{k}_3 = 1, \widehat{k}_7 = 1, \widehat{k}_{10} = 0, \widehat{k}_{12} = 1$.

Set Intersection Phase

1. The server S creates $F = (f_1 = 1, f_2 = 0, f_3 = 1, f_4 = 1, f_5 = 0, f_6 = 0, f_7 = 1, f_8 = 0, f_9 = 1, f_{10} = 0, f_{11} = 1, f_{12} = 0, f_{13} = 0 f_{14} = 1, f_{15} = 1, f_{16} = 0, f_{17} = 0, f_{18} = 0, f_{19} = 0, f_{20} = 0, f_{21} = 0, f_{22} = 0)$. It then computes $G = F \oplus Key_3 = (f_1 \oplus \widehat{k}_1, \ldots, f_{22} \oplus \widehat{k}_{22}) = (g_1, \ldots, g_{22})$ sends G to C. Note that $g_3 = 1 \oplus 1 = 0$, $g_7 = 1 \oplus 1 = 0$, $g_{10} = 0 \oplus 0 = 0$ and $g_{12} = 0 \oplus 1 = 1$ since $\widehat{k}_3 = 1, \widehat{k}_7 = 1, \widehat{k}_{10} = 0, \widehat{k}_{12} = 1$.
2. Note that the client knows only $\widehat{k}_3 = 1, \widehat{k}_7 = 1, \widehat{k}_{10} = 0, \widehat{k}_{12} = 1$. The client C, on receiving $G = (g_1, \ldots, g_{22})$, computes $h_1 = g_{\psi(p_1)} \oplus \widehat{k}_{\psi(p_1)} = g_7 \oplus \widehat{k}_7 = 0 \oplus 1 = 1$, $h_2 = g_{\psi(p_2)} \oplus \widehat{k}_{\psi(p_2)} = g_{10} \oplus \widehat{k}_{10} = 0 \oplus 0 = 0$, $h_3 = g_{\psi(p_3)} \oplus \widehat{k}_{\psi(p_3)} = g_3 \oplus \widehat{k}_3 = 0 \oplus 1 = 1$ and $h_4 = g_{\psi(p_4)} \oplus \widehat{k}_{\psi(p_4)} = g_{12} \oplus \widehat{k}_{12} = 1 \oplus 1 = 0$. Since $h_1 = 1$ and $h_3 = 1$, therefore $\xi = \{\psi(p_1), \psi(p_3)\} = \{\psi(1), \psi(8)\} = \{7, 3\} = \{3, 7\}$ is $X \cap Y$

References

1. Abadi, A., Terzis, S., Dong, C.: O-psi: delegated private set intersection on outsourced datasets. In IFIP International Information Security Conference, pp. 3–17. Springer, Berlin (2015)
2. Abadi, A., Terzis, S., Dong, C.: Vd-psi: verifiable delegated private set intersection on outsourced private datasets. Finanacial Cryptography and Data Security (2016)
3. Abadi, A., Terzis, S., Metere, R., Dong, C.: Efficient delegated private set intersection on outsourced private datasets. IEEE Trans. Dependable Secure Comput. (2017)
4. Cheng, X., Guo, R., Chen, Y.: Cryptanalysis and improvement of a quantum private set intersection protocol. Quantum Inf. Process. **16**(2), 37 (2017)
5. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In Advances in Cryptology-EUROCRYPT 2004, pp. 1–19. Springer, Berlin (2004)
6. Fuchs, C.A.: Distinguishability and accessible information in quantum theory. arXiv preprint arXiv:quant-ph/9601020 (1996)
7. Gao, F., Liu, B., Wen, Q.-Y., Chen, H.: Flexible quantum private queries based on quantum key distribution. Opt. Express **20**(16), 17411–17420 (2012)
8. Hazay, C., Venkitasubramaniam, M.: Scalable multi-party private set-intersection. In IACR International Workshop on Public Key Cryptography, pp. 175–203. Springer, Berlin (2017)
9. Helstrom, C.W., Helstrom, C.W.: Quantum detection and estimation theory, vol. 3. Academic press, New York (1976)
10. Herzog, U., Bergou, J.A.: Optimum unambiguous discrimination of two mixed quantum states. Phys. Rev. A **71**(5), 050301 (2005)
11. Inbar, R., Omri, E., Pinkas, B.: Efficient scalable multiparty private set-intersection via garbled bloom filters. In International Conference on Security and Cryptography for Networks, pp. 235–252. Springer, Berlin (2018)
12. Jakobi, M., Simon, C., Gisin, N., Bancal, J.-D., Branciard, C., Walenta, N., Zbinden, H.: Practical private database queries based on a quantum-key-distribution protocol. Phys. Rev. A **83**(2), 022301 (2011)
13. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.: Scaling private set intersection to billion-element sets. In International Conference on Financial Cryptography and Data Security, pp. 195–215. Springer, Berlin (2014)
14. Kavousi, A., Mohajeri, J., Salmasizadeh, M.: Improved secure efficient delegated private set intersection. arXiv preprint arXiv:2004.03976 (2020)
15. Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., Trieu, N.: Practical multi-party private set intersection from symmetric-key techniques. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1257–1272. ACM, (2017)
16. Li, F., Niu, B., Wang, Y. et al.: Server-aided private set intersection based on reputation. Information Sciences (2016)

17. Maitra, A.: Quantum secure two-party computation for set intersection with rational players. Quantum Inf. Process. **17**(8), 197 (2018)
18. Miyaji, A., Nishida, S.: A scalable multiparty private set intersection. In International Conference on Network and System Security, pp. 376–385. Springer, Berlin (2015)
19. Raynal, P.: Unambiguous state discrimination of two density matrices in quantum information theory. arXiv preprint arXiv:quant-ph/0611133 (2006)
20. Shi, R., Yi, M., Zhong, H., Cui, J., Zhang, S.: Two quantum protocols for oblivious set-member decision problem. Sci. Rep. **5**, 15914 (2015)
21. Shi, R., Yi, M., Zhong, H., Cui, J., Zhang, S.: An efficient quantum scheme for private set intersection. Quantum Inf. Process. **15**(1), 363–371 (2016)
22. Shi, R., Yi, M., Zhong, H., Zhang, S.: Quantum oblivious set-member decision protocol. Phys. Rev. A **92**(2), 022309 (2015)
23. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
24. Wang, Q., Zhou, F., Xu, J., Peng, S.: Tag-based verifiable delegated set intersection over outsourced private datasets. IEEE Trans. Cloud Comput. (2020)
25. Yang, X., Luo, X., Wang, X.A., Zhang, S.: Improved outsourced private set intersection protocol based on polynomial interpolation. Concurr. Comput. Pract. Exp. **30**(1), e4329 (2018)
26. Zhang, E., Jin, G.: Cloud outsourcing multiparty private set intersection protocol based on homomorphic encryption and bloom filter. J. Comput. Appl. (8):20 (2018)
27. Zhang, E., Li, F., Niu, B., Wang, Y.: Server-aided private set intersection based on reputation. Inf. Sci. **387**, 180–194 (2017)
28. Zhang, E., Liu, F.-H., Lai, Q., Jin, G., Li, Y.: Efficient multi-party private set intersection against malicious adversaries. In Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, pp. 93–104, (2019)