PSKPIR: Symmetric Keyword Private Information Retrieval based on PSI with Payload*

Zuodong Wu$^{1,2[0000-0001-7702-4965]}$, Dawei Zhang$^{1,2[0000-0001-5942-8245]}$, Yong Li$^{3[0000-0002-1419-6257]}$, and Xu Han$^{1,2[0000-0002-1030-2462]}$

1 School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
2 Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing, China dwzhang@bjtu.edu.cn
http://scit.bjtu.edu.cn
3 School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China
liyong@bjtu.edu.cn

Abstract. Symmetric Private Information Retrieval (SPIR) is a protocol that protects privacy during data transmission. However, the existing SPIR focuses only on the privacy of the data to be requested on the server, without considering practical factors such as the payload that may be present during data transmission. This could seriously prevent SPIR from being applied to many complex data scenarios and hinder its further expansion. To solve such problems, we propose a primitive (PSKPIR) for symmetric private keyword information retrieval based on private set intersection (PSI) that supports payload transmission and batch keyword search. Specifically, we combine probe-and-XOR of strings (PaXoS) and Oblivious Programmable PRF (OPPRF) to construct PSI with payload (PSI-Payload) not only satisfies client privacy and server privacy, but also facilitates efficient payload transmission. The client can efficiently generate symmetric keys locally using keywords in the intersection, and receive payloads with matching labels in batches. In addition, we provide security definitions for PSKPIR and use the framework of universal composability (UC) to prove security. Finally, we implement PSKPIR with sublinear communication costs in both LAN and WAN settings. Experimental results show that our payload transfer speed is 10× faster than previous work on sufficiently large data sets.

Keywords: Symmetric private information retrieval · Private information retrieval · Private set intersection with payload · Batch retrieval · Payload data transfer.

1 Introduction

The Private Information Retreival (PIR) [1,2,8,30,33] protocol considers the client's privacy and allows them to retrieve data from the server while hiding data of interest to them. However, some servers may contain valuable data

* Supported by National Natural Science Foundation of China (Grant no.U21A20463).

(e.g., medical servers, government document servers), and it is necessary for the PIR protocol to protect the server's privacy as well. For this question, Gertner et al. [13] proposed the Symmetric Private Information Retrieval (SPIR) protocol to enable clients to retrieve data without revealing any information about the server except for the requested data. This approach ensures the privacy of the client retrieval and maintains the confidentiality of the server's data. These properties have made SPIR one of the most promising candidate protocols for PIR extension protocol research. [18,28,31,32]. However, most these implemented SPIR protocol focuses primarily on ensuring privacy for both parties while frequently overlooking the issue of batch item retrieval and payload data transfer [21,12,14]. In many instances, the server must not only supply privacy data for the specific item requested by the client, but for complex data, the client may also need to provide additional payload data (data size, check bit), which also requires privacy guarantees in the SPIR protocol. In addition, considering the vast and personalized data requirements of client, SPIR also incorporate batch data retrieval to deliver efficient and precise retrieval services. In order to explain the above two types of requirements, we start with envisaged scenario:

Scenario 1. When a well-known international investment institution seeks to invest in the driverless industry, it relies on securities companies to inquire about and purchase stocks related to the industry. In order to prevent competitors from learning about their intentions and causing volatility in the stock market, the investment institution expects securities companies to protect the anonymity of their identities as well as their retrieval records and trading data. In turn, securities companies need to protect their clients' valuable information from unlawful access. The standard SPIR protocol provides a viable solution for meeting these requirements.

However, in practice, investment internationals require background information for analysis when purchasing stocks of driverless technology companies through securities companies, such as audit reports, public opinion monitoring, shareholder changes, and other relevant information to assist in subsequent investment decisions. Furthermore, since there are many companies in the autonomous driving industry, it is necessary for investment companies to search for multiple autonomous driving companies in batches in order to obtain better analysis results.

Privacy Set Intersection (PSI), as a mainstream privacy computing technology [6,7,9,10,11,15,20,25,26,27], can effectively address the security and functional requirements of Scenario 1 [11,15,23]. The PSI allows the client to get the intersection data of both parties, and the server does not need to worry about leaking data outside the intersection. Therefore, PSI can not only ensure the privacy of both parties, but can also be applied to PIR and SPIR, where both parties can calculate the intersection of two private data sets to form a redirected authentication or negotiation process, which is conducive to secure data transmission [29]. Besides, Pinkas et al. [23] introduced a new variant, PSI-Payload, where each keyword is associated with some data ("payload"). Although the purpose of this protocol is to calculate payload data rather than transmit it, PSI-Payload provides a promising example for building a SPIR protocol that meets the requirements of scenario 1 because of its unique advantages. Privacy-preserving Policy-based Information Transfer (PPIT) is a Privacy-preserving Set Intersection (PSI) variant proposed by De Cristofaro et al. [5]. It comprises three protocols: RSA-PPIT, Schnorr-PPIT, and IBE-PPIT. These protocols can not only meet the privacy requirements of both parties involved, but also allow client to retrieve data in batches. However, all PPIT incur expensive computational costs for authentication.

Goals and Challenges. In this paper, we aim to introduce PSI-Payload for the development of a SPIR protocol that meets the practical requirements of scenario 1, and then propose a new primitive (PSKPIR) to improve batch retrieval and payload data transmission capabilities. However, in specific cases, some challenges also need us to solve. Firstly, most protocols such as Jarecki et al. [15] and Pinkas et al. [23] are based on public key or OT, which may lead to increased computing or communication costs when using PSI-Paylaod to transmit payload data. Thus, it is necessary to identify a PSI technique that strikes a balance between data privacy and transmission efficiency. Secondly, the design of PSKPIR should align with the privacy requirements of SPIR and the practical needs of scenario 1. Thirdly, our protocol can achieve malicious security with no significant increase in overhead compared to the semi-honest setting.

1.1 Our Contributions

Specifically, our main contributions are summarized as follows:

- **A new primitive**. First, we propose a new primitive built through PSI-Payload, PSKPIR, to support batch keyword retrieval and improve payload data transmission capabilities. Our PSI-Payload is built on PaXOS and OP-PRF, where PaXoS can not only compress keywords into vectors in batches, but also improve the stability of encoding and transmission efficiency. In addition, the OPPRF function based on Vector-OLE can achieve sublinear communication consumption, and the resulting PSKPIR can achieve stronger security with lower performance costs. The client can not only calculate the negotiated key locally by using keywords in the intersection, but also realize batch retrieval of payload data through matching labels.
- **Formal definition and malicious setting**. Secondly, we formalize the definition of the PSKPIR protocol and show that PSKPIR can be constructed under the semi-honest and malicious model. Security analysis shows that PSKPIR can effectively protect the privacy of both parties under the UC framework.
- **Lower communication and higher speed**. Finally, we demonstrate the feasibility of the proposed PSKPIR protocol with experimental results. Compared to PPIT, our protocol requires much less than factor 10 communication. Our protocol improves runtime by up to a factor of 20 in the WAN setting and up to a factor of 10 in the LAN setting.

2 Related Work

2.1 PIR

Chor et al. [4] first proposed a computationally secure PIR protocol based on a single function and realized the privacy of the clinet, but its communication complexity is $o(n^2)$. In 2005, Lipmaa et al. [19] proposed an efficient single-server PIR protocol. The protocol uses homomorphic encryption and oblivious transfer (OT) to ensure client privacy, and the communication complexity is $o(\log_2 n)$. In order to further reduce the computational overhead of PIR, Giovanni et al. [8] demonstrate the equivalence between PIR protocol and OT. They also use a variant of OT to build the PIR protocol, so that the security of the PIR protocol no longer requires weak computing assumptions [24]. Up to now, most PIR protocols do not consider the privacy issue of the server, but focus more on reducing the calculation and communication consumption of the protocol.

2.2 SPIR

Aiming at the problem that the PIR protocol cannot guarantee server privacy, Gertner et al. [12] implements an honest client SPIR protocol, achieving privacy for both parties. However, the protocol structure is complex and makes it difficult to implement data transfer. In 2013, Jarecki et al. [14] proposed an outsourced SPIR protocol. The protocol is designed to facilitate arbitrary boolean queries in malicious model, while at the same time withstanding adversarial non-colluding servers. Inspired by literature [8], Naor et al. [21] demonstrated how to efficiently and directly transform any PIR protocol into a SPIR protocol. Moreover, the SPIR protocol built by the OT protocol can have the advantage of sublinear communication. This greatly reduces the communication cost of SPIR protocol in data transmission.

3 Preliminaries

Below we present pivotal concepts and technologies that are fundamental to the comprehension of the PSKPIR.

3.1 Notations

Let n denote the set of natural numbers. The expression $x \leftarrow M$ signifies that x is chosen uniformly at random from the set M. The notation $y \leftarrow \mathbb{A}(x)$ represents the probability that algorithm $\mathbb{A}$ outputs y given input x. The probability of event $\mathbb{E}$ occurring after undergoing random processes $r_1, \ldots, r_n$ is denoted as $\Pr[r_1; \ldots; r_n : \mathbb{E}]$. The notation $\{\mathrm{R}_1; \ldots; \mathrm{R}_\mathrm{m}\text{: v}\}$ refers to a sequence of random processes that yield the value v. The function $t_{C,S}(x, r_C, y, r_S)$ represents the transcript of an execution of an interactive protocol, where party C inputs x and party S inputs y, along with their respective random strings r_C and r_S.

The expression $(r_C, r_S, t) \leftarrow t_{C,S}(x, \cdot, y, \cdot)$ denotes the output of algorithm $\mathbb{A}$ after the execution, with r_C and r_S generated randomly. Similarly, $(r_S, t) \leftarrow t_{C,S}(x, r_C, y, \cdot)$ represents the process of selecting a random string r_S and setting $t = t_{C,S}(x, r_C, y, r_S)$. Likewise, $(r_C, t) \leftarrow t_{C,S}(x, \cdot, y, r_S)$ signifies the process of selecting a string r_C, and $(r_C, r_S, t) \leftarrow t_{C,S}(x, \cdot, y, \cdot)$ indicates that r_S and r_C are uniformly chosen at random. The notation $\overrightarrow{A} = (a_1, \ldots, a_n)$ represents row vectors, and $M_{i,j}$ denotes the element at row i and column j of matrix M. The expression $< \overrightarrow{A}, \overrightarrow{B} >$ denotes the inner product of vectors $\overrightarrow{A}$ and $\overrightarrow{B}$.

3.2 PaXoS

PaXoS is a mapping function with excellent linear characteristics [27]. In the present study, PaXoS is regarded as a Garbled Cuckoo Table function. The function entails mapping binary strings to a binary vector with length of m while simultaneously preserving the linear independence of the output binary vector. PaXoS is agnostic to any input and guarantees that no input information is leaked.

Definition 1. *Let (n, m, ε) is a random algorithm $v^{\mathcal{H}} : \{0,1\}^* \rightarrow \{0,1\}^m$, we say that (n, m, ε) is a PaXoS algorithm for any $x_1, \ldots, x_{\mathrm{n}} \in \{0,1\}^*$ if there $a_1, \ldots, a_n$ can be arbitrary, not all 0, such that:*

$$\text{Prob}[a_1 v^{\mathcal{H}}(x_1), \ldots, +a_n v^{\mathcal{H}}(x_n) = 0] \geq 1 - 1/2^{\varepsilon} \tag{1}$$

where ε is sufficiently large, and $v^{\mathcal{H}}(x)$ is a random vector which independence is over the vector space $(\mathcal{Z}_2)^m$.

Definition 2. *Let (n, m, ε) is a random algorithm $v^{\mathcal{H}} : \{0,1\}^* \rightarrow \{0,1\}^m$, we say that (n, m, ε) is a PaXoS algorithm for any $x_1, \ldots, x_{\mathrm{n}} \in \{0,1\}^*$ if there $a_1, \ldots, a_n$ can be arbitrary, not all 0, such that:*

$$\text{Prob}[a_1 v^{\mathcal{H}}(x_1), \ldots, +a_n v^{\mathcal{H}}(x_n) = 0] \geq 1 - 1/2^{\varepsilon} \tag{2}$$

where ε is sufficiently large, and $v^{\mathcal{H}}(x)$ is a random vector which independence is over the vector space $(\mathcal{Z}_2)^m$.

Garbled Cuckoo Table. Our construction utilizes Garbled Cuckoo Tables, which are designed to allow the data to exhibit near-optimal size and optimal encoding/decoding speed during the preprocessing [22]. The construction of the Garbled Cuckoo Table combines both the garbled Bloom filter (GBF) and Cuckoo hashing, but has unique advantages over them: 1) Parameter: It has just 2 hash functions instead of λ and Cuckoo hashing and can be used against malicious models; 2) Speed: The ratio is fixed at n/m with a constant rate, while the GBF has a rate of $O(1/\lambda)$: 3) Private: The independence of mapping from inputs is implicit.

3.3 Vector-OLE (VOLE)

Vector-OLE [3] represents a valuable extension to the Oblivious Linear Function Evaluation (OLE) method, which allows the Client to obtain information about any linear combination of two vectors owned by the Server.

VOLE Functionality. The figure 1 depicts the VOLE ideal function denoted as F_{vole}. The Server obtains a random value Δ and a random vector $\overrightarrow{B}$ of length m from the field $\mathcal{F}_m$. Similarly, the Client receives a random vector $\overrightarrow{A}$ from $\mathcal{F}_m$ and the vector $\overrightarrow{C} = \overrightarrow{B} + \Delta\overrightarrow{A}$.

VOLE Ideal Functionality F_{vole}

Parameters:
$\mathcal{F}$: a random field.
Functionality 1:
(No Corrupted): The $\overrightarrow{A}$ and $\overrightarrow{B}$ is sample from $\mathcal{F}$ and define $\overrightarrow{C} := \overrightarrow{B} + \overrightarrow{A}\Delta$. If the functionality does not reply success, abort.
(Corrupted Client): The Δ is a sample from the F. Then the functionality computes $\overrightarrow{B} = \overrightarrow{C} - \overrightarrow{A}\Delta$, where the $\overrightarrow{A}$ and $\overrightarrow{C}$ are sent from the Client.
(Corrupted Server): The $\overrightarrow{A}$ is a sample from the $\mathcal{F}$. Then the functionality computes $\overrightarrow{C} := \overrightarrow{B} + \overrightarrow{A}\Delta$, where the Δ and $\overrightarrow{B}$ are sent from the Client.
Outputs:The functionality sends $\overrightarrow{B}$, Δ to the Server and $\overrightarrow{A}$, $\overrightarrow{C}$ to the Client.

Fig. 1: Ideal functionality F_{vole} of VOLE.

3.4 Oblivious Programmable PRF (OPPRF)

The OPPRF constructed is largely based on Oblivious Pseudorandom Function (OPRF) [17], and OPPRF has the same security as OPRF. In this section, we first review the basic functionality of OPRF and then introduce its programmable variant: OPPRF.

OPRF. OPRF is a protocol for calculating PRF covertly, and its output results are random and unpredictable [16]. The Server holds the key k, and the Client holds the input x. The two parties jointly calculate the function $F_k(x)$ through interaction, and finally the Client gets the function value. We describe the ideal functionality for an OPRF in Figure 2.

OPPRF. Compared to OPRF, OPPRF adds programmable properties. The server may set the PPF output at certain points, and these points and the PPF output may be determined by the server. The ideal functionality for an OPPRF is shown in Figure 3.

3.5 Private Set Intersection with Payload (PSI-Payload)

PSI-Payload [15] is a cryptographic protocol that allows two parties, Client and Server, to privately compute the intersection of their sets while also including an

OPRF Ideal Functionality F_{OPRF}

Parameters:
F_r: some PRF family. n: the size of set X, Y.
Functionality 1:
(No Corrupted): Wait for the Client send a set of $X = \{x_1, \ldots, x_n\}$. The functionality samples the $F_r : \{0,1\}^* \to \{0,1\}^l$, where the key r is uniformly chosen.
(Corrupted Client): Wait for the Client send a set of $X' = \{x_1, \ldots, x_{n'}\}$, where $n' \neq n$. The functionality samples the $F_r : \{0,1\}^* \to \{0,1\}^l$ and sends $\{F_r(x_1), \ldots, F_r(x_{n'})\}$ to the Client.
Outputs: the functionality returns $F_r(y)$ to the Server.

Fig. 2: Ideal functionality F_{OPRF} of OPRF

OPPRF Ideal Functionality F_{OPPRF}

Parameters:
Y: a set $\mathrm{Y} = \{(\mathrm{y}_1,\tilde{\mathrm{y}}_1), \ldots, (\mathrm{y}_n,\tilde{\mathrm{y}}_n)\}$ with n sets of data, where $\mathrm{y_i} \in \{0,1\}^*$ and $\tilde{\mathrm{y}}_i \in \{0,1\}^{\mathrm{l}}$. l:the output bit length.
Functionality 1:
Wait for the Client send a set of $X = \{x_1, \ldots, x_n\}$. The functionality samples the $F_k :\to \{0,1\}^l$, where the key k is uniformly chosen and for each $(\mathrm{y}_i,\tilde{\mathrm{y}}_i) \in \mathrm{Y}$, satisfying $F_k(y_i) = \tilde{y}_i$. Then the functionality sends set of $X' = \{F_k(x_i)|x_i \in X\}$ to the Client.
Outputs:the functionality returns $F_r(y)$ to the Server.

Fig. 3: Ideal functionality F_{OPPRF} of Oblivious Programmable PRF

associated data (payload) for each element in the intersection. The PSI-Payload ideal functionality is presented in Figure 4.

PSI-Payload Functionality. The PSI-Payload protocol uses intersection operations to facilitate the safe computation of payload data, while also ensuring that both parties'privacy is maintained. This protocol implements anonymous privacy calculation of payload data while maintaining the confidentiality of their respective private request retrieval items.

4 PSKPIR

Informally, our PSKPIR is an interactive protocol between two parties, the S (Server) and the C (Client). The S holds the target keyword ID' and payload data $D \in \{0,1\}^n$, where $n \in N$. The C holds retrieval keyword ID'. First, the Client sends multiple keywords ID for retrieval (random algorithm generates input). Then the server performs keyword matching (efficient PSI-Payload). Finally, the client can obtain the corresponding payload data for their retrieval keywords (symmetric encryption algorithm).

PSI-Payload Ideal Functionality $F_{PSI}^{Payload}$

Parameters:

Y: a set $\mathrm{Y} = \{(\mathrm{y}_1,\tilde{\mathrm{y}}_1), \ldots, (\mathrm{y}_n,\tilde{\mathrm{y}}_n)\}$ with n sets of data, where $\tilde{\mathrm{y}}_i$ is a payload data.

Functionality:

(No Corrupted): Wait for the Client send a set of $X = \{x_1, \ldots, x_n\}$ and the Server send a set of $\mathrm{Y} = \{(\mathrm{y}_1,\tilde{\mathrm{y}}_1), \ldots, (\mathrm{y}_n,\tilde{\mathrm{y}}_n)\}$.

(Corrupted Client): If the Client send a set of X, where the $|X| > n$, the functionality abort.

(Corrupted Server): If the Server send a set of Y, where the $|Y| > n$, the functionality abort.

Outputs: The functionality returns $\widetilde{X} = \{(x_i, \widetilde{y_i})|x_i = y_i\}$ to the Client.

Fig. 4: Ideal functionality $F_{PSI}^{Payload}$ of PSI-Payload

4.1 The Formal Definitions

Next, we will focus on formally defining a PSKPIR protocol in the two-party setting. Let (C, S) be an interactive protocol, and let $\mathcal{F}$ be a polynomial time algorithm. We define that (C, S, F) is a PSKPIR protocol if:

Definition 3 (Correctness). *For each $ID_i, 1 \leq i \leq n$ and $D = \{D_i \in \{0,1\}^n | 0 \leq i \leq n\}$ from Server, each $ID_i, 1 \leq i \leq n$ from Client, and all constants c, and all sufficiently large k,*

$$\begin{aligned}&\Pr[(r_C, r_S, t) \leftarrow t_{C,S}((1^k, ID', D), \cdot, (1^k, n, ID),\cdot)\\ &: F(1^k, r_S, ID, t, n) = D] \geq 1 - k^{-c}\end{aligned} \tag{3}$$

Definition 4 (Client Privacy). *For each $i, j \in \{1, \ldots, n\}$, each $ID_i, 1 \leq i \leq n$ and $D = \{D_i \in \{0,1\}^n | 0 \leq i \leq n\}$ from Server, and each $ID_i, 1 \leq i \leq n$ for Client, each polynomial time S', for all constants c, and all sufficiently large k, it holds that $|P_i - P_j| \leq k^{-c}$, where*

$$\begin{aligned}&P_i = \Pr[(r_{S'}, r_C, t) \leftarrow t_{S',C}((1^k, ID'), \cdot, (1^k, n, ID),\cdot) :\\ &S'(1^k, ID_i, r_{S'}, t, n) = 1]\\ &P_j = \Pr[(r_{S'}, r_C, t) \leftarrow t_{S',C}((1^k, ID'), \cdot, (1^k, n, ID),\cdot) :\\ &S'(1^k, ID_j, r_{S'}, t, n) = 1]\end{aligned} \tag{4}$$

Definition 5 (Server/Data Privacy). *For each polynomial time C', each random string $r_{c'}$, there exists two set $D = \{D_i | 1 \leq i \leq n\}, D' = \{D'_i | 1 \leq i \leq n\}$ and such that D_i=D'_i. For all sufficiently large k, for all constants c, it holds that $|P_D - P_{D'}| \leq k^{-c}$, where*

$$\begin{aligned}P_D &= \Pr[(r_{C'}, r_S, t) \leftarrow t_{S,C'}((1^k, ID', D), \cdot, (1^k, n, ID), \cdot)\\ &\quad : C'(1^k, n, ID_i, r_{C'}, D, t) = 1]\\ P_{D'} &= \Pr[(r_{C'}, r_S, t) \leftarrow t_{S,C}((1^k, ID', D'), \cdot, (1^k, n, ID), \cdot)\\ &\quad : C'(1^k, n, ID_i, r_{C'}, D', t) = 1]\end{aligned} \tag{5}$$

5 Main Construction

This section introduces the components and detailed content of PSKPIR in a semi-honest setting. Without loss of generality, we assume that ID instead of i is the keyword retrieved by the Client from the Server. The Server will transmit payload data to the Client according to the ID.

5.1 Components

The PSKPIR involves two players:

- **Server:** stores the set $ID' = \left\{ID'_i \in \{0,1\}^l, 0 \leq i \leq n\right\}$. The ID'_i as an keyword uniquely identifies the payload data D, where $D_i \in \{0,1\}^*$.
- **Client:** set of keywords $ID = \{ID_i | ID_i \in \{0,1\}^l, 0 \leq i \leq \text{n}\}$, where ID_i is an l-bit string.

5.2 PSKPIR with Semi-Honest Model

Theorem 1 (correctness). *In the F_{vole}, $F_{PSI}^{payload}$-hybrid model with a random oracle, PSKPIR implements correctness with computational security in Semi-Honest environment, corresponding to Definition 3.*

Proof. Assuming $\overrightarrow{p}$ is a non-trivial random solution vector, distinct from the zero vector $(0, 0, \ldots, 0)$, the protocol is advanced by the Client transmitting $\overrightarrow{A} + \overrightarrow{p}$ to the Server. Upon receiving this transmission, the Server proceeds to define a vector $\overrightarrow{K}$ as $\overrightarrow{K} = \Delta(\overrightarrow{A} + \overrightarrow{p}) + \overrightarrow{B}$. The crucial observation is that

$$\begin{aligned} M\overrightarrow{K} &= M\overrightarrow{B}^T + \Delta(M\overrightarrow{A} + M\overrightarrow{p}^T) \\ &= M\overrightarrow{B}^T + \Delta M\overrightarrow{A}^T \\ &= M\overrightarrow{C}^T \end{aligned} \tag{6}$$

For each ID'_i, the Server holds that $< \overrightarrow{M_s}, \overrightarrow{K} >=< \overrightarrow{M_c}, \overrightarrow{C} >$ where $\overrightarrow{M_s}$ is the ID'_i's row of M_s. An OPRF can then be obtained by having the Client apply a random oracle as $H_1(< \overrightarrow{M_c}, \overrightarrow{C} >)$ while the server computes the output at any ID'_i as $F_{\overrightarrow{K}}(ID') := H_1(< \overrightarrow{M_s}, \overrightarrow{K} >)$.

The second building block enables the Server to sample an OPPRF key, denoted as $\overrightarrow{\text{K}}$, such that the output of the function $F_{\overrightarrow{K}}(ID'_i)$ is equal to the desired value v_i for a particular ID'_i selected by the Server. In contrast, $F_{\overrightarrow{K}}$ returns a random output at all other data points. The Server solves the system $M_s\overrightarrow{p'}$, observe that at $ID_i = ID'_i \in ID'$:

$$\begin{aligned} x' &= F'(ID_i) + < \overrightarrow{M^*{}_c}, \overrightarrow{p}' > \\ &= F'(ID_i) + v_i - F'(ID_i) \\ &= v_i \end{aligned} \tag{7}$$

PSKPIR Protocol with Semi-Honest Model

Parameters:
$H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$, $H_2 : \{0,1\}^* \to \{0,1\}^{l_2}$: two hash functions. $v = \{v_1, \ldots, v_n\}$: random variables set. X: a set $X = \{ID_i, \ldots, ID_n\}$ with n of keyword. Y: a set $\mathrm{Y} = \{(\mathrm{ID'}_1,\mathrm{D}_1), \ldots, (\mathrm{ID'}_n,\mathrm{D}_n)\}$ with n sets of data, where the ID'_i as a keyword and $D_i \in \{0,1\}^{l_1}$ is payload data.
Input: Input $(start, sid, X)$ from the Client and $(start, sid, Y)$ from the Server.
Setup:

1. The Client sends $(Client, sid)$ and the Server sends $(Client, sid)$ to F_{vole}. Then the Client receives $\overrightarrow{C} = \Delta\overrightarrow{A} + \overrightarrow{B}$, $\overrightarrow{A}$ and the Server receives Δ, $\overrightarrow{B}$.
2. The Client takes $(ID_1, ID_2, \ldots, ID_n)$ map the random binary matrix $M_c = [v^H(ID_1), \ldots, v^H(ID_n)]^T$ by PaXoS. The Client then solves the linear system: $M_C\overrightarrow{p} = (0, 0, \ldots, 0)^T$.

PSI-Payload:

3. The protocol proceeds by having the Client sends $\overrightarrow{A} + \overrightarrow{p}$ and r to the Server that defines $\overrightarrow{K} = \Delta(\overrightarrow{A} + \overrightarrow{p}) + \overrightarrow{B}$. Meanwhile, the Server outputs $M_s = [\mathrm{v^H}(ID'_1, r), \ldots, \mathrm{V^H}(ID'_n, r)]^T$.
4. The parties perform an OPRF protocol and the Server outputs the OPRF function is $F_{\overrightarrow{K}}(ID'_1) = H_1(Decode(\overrightarrow{M_S^*}, \overrightarrow{K})) = H_1(Decode(\overrightarrow{M_C^*}, \overrightarrow{C}))$.
5. The server utilizes the PaXoS solver to find the solution to the linear system: $M_s\overrightarrow{p'} = (v_1 - F'(ID'_1), v_2 - F'(ID'_2), \ldots, v_n - F'(ID'_n))^T$ and sends $\overrightarrow{p'}$ to the Client.

Transfer:

6. The Server encrypts plaintext D_i using symmetric secret key: $Enc_{H_1(v_i)}(D_i) = C_i, i \in \{1, n\}$.
7. Then the Server computes the labels: $t_i = H_2(H_1(v_i)), i \in \{1, n\}$, and sends ciphertext sets and label sets $\overrightarrow{C} = \{C_i\}$, $\overrightarrow{T} = \{t_i\}$ to the Client.
8. The Client computes $x_i = F'(ID_i) + < \overrightarrow{M_c}, \overrightarrow{p'} >, i \in \{1, n\}$. If $ID_i = ID'_i$, then $x_i = v_i$ as intended. Otherwise, the output will be unpredictable.
9. The Client computes and outputs $t'_i = H_2(H_1(x_i)), i \in \{1, n\}$.

Outputs: If $\exists i \in \{1, n\}, t'_i = t_i$, the Client decrypts and outputs $D'_i = Dec_{x_i}(C_i), i \in \{1, ...n\}$.

Fig. 5: PSKPIR Protocol in Semi-Honest Secure Setting

To prove the correctness of the PSKPIR protocol, we also need to show that the Client outputs $\{(ID_i, D_i)|ID_i = ID'_i\}$ with a probability of at least $1 - k^{-c}$. First, note that if the Client is able to correctly transfer payload data $\{D_i, 1 \leq i \leq n\}$ after executing the PSI-Payload protocol in step 5, then it can compute the correct $\{D_i, ID' \in X\}$ in step 8. Next, based on the randomness and unpredictability of the OPRF/OPPRF, the Client outputs $\{D_i|ID_i = ID'_i\}$ with probability at least $(1 - k^{-c})^{|ID_i = ID'_i, 1 \leq i \leq n|}$ since PSKPIR calculates $|ID \cap ID'|$ and transfers $\{D_i|ID_i \in X\}$ are all independent.

Theorem 2 (Client Privacy). *In the F_{OPRF},F_{vole},$F_{PSI}^{payload}$-hybrid model, the protocol in Figure 5 can ensure client privacy in Semi-Honest environment, according to definition 4.*

Proof. It is assumed that there exists a probabilistic polynomial-time algorithm that can guess $\{ID_i|1 \leq i \leq \mathrm{n}\}$ with a probability of at least k^{-c}, for some constant c and infinitely many k, after running the protocol. In step 4, the Client sends $\overrightarrow{\mathrm{A}} + \overrightarrow{\mathrm{p}}$ to the Server, such that $\overrightarrow{p}$ is the solution of the linear system. In addition, $\overrightarrow{\mathrm{A}} + \overrightarrow{\mathrm{p}}$ is evenly distributed for the semi-honest Server. The parties then perform OPRF protocol under PSI-Payload invocations. Therefore, the semi-honest Server can guess $\{ID_i|1 \leq i \leq \mathrm{n}\}$ with at least k^{-c} probability. This implies, by a hybrid argument, that for some keyword $\{ID_i|1 \leq i \leq \mathrm{n}\}$, the Server can guess with probability at least $1/n + k^{-c}$. Since all maps $v^H(\cdot)$ are linear and independent (assuming that the random binary of different keywords is linear and independent), it is straightforward to use the Server to construct a random binary map that distinguishes in a PSI-Payload execution between the different keywords, with a probability of at least $1/n+k^{-c}$. Since n is polynomial, this is a non-negligible advantage, and thus contradicts our assumption and indirectly verifies the correctness of the client privacy.

Theorem 3 (Server Privacy). *In the F_{OPRF}, F_{OPPRF}, and $F_{PSI}^{payload}$-hybrid model, our protocol is server/Data privacy in Semi-Honest environment according to definition 5.*

Proof. We first assume that there exists a semi-honest Client that can guess $\{D_i|ID_i \notin ID'\}$ with probability ε in the probability polynomial algorithm. In step $3-5$, the Server defines the OPPRF protocol. The outside of the intersection is theoretically hidden data. Moreover, since OPRF exhibits the property of randomness, it can be shown that at any other point in the input domain where ID_i does not belong to the subset ID', the output is completely random. Therefore, the scenario where ID_i is not equal to ID'_i and $F'(ID_i) = F'(ID'_i)$ is unattainable. In the semi-honest setting, the OPRF approach is to define the output domain of F' to be $\{0,1\}^{out}$ where $out := \lambda + \log_2{}^{(2n)}$, the overall probability of a collision is $2^{-\lambda + \log_2{}^{(2n)}}$. Then, in step 9, the Client invokes two random oracles H_1, H_2 are independent to output t'_i and determines $t'_i = t_i$. This means that the Client can guess $\{D_i|ID_i \notin ID'\}$ with probability $\varepsilon = 2^{(\lambda - \log_2{}^{(2n)})l_1 l_2}$, where the probability of H_1 collision is 2^{-l_1} and the probability of H_2 collision is

2^{-l_2}. Since λ, l_1, l_2 are polynomial and $2^{(\lambda - \log_2{}^{(2n)})l_1 l_2}$ are negligible, the protocol guarantees Server/data privacy.

6 The Enhanced Protocol

In this section, we present a PSKPIR protocol that is secure against a malicious adversary, and Appendix A provides proof of its security under the UC model. One security concern is that $\overrightarrow{p}$ could leak the data about ID. Our protocols are optimized with XoPaXoS (X-oblivious PaXoS) [27]. In our malicious model construction, we will use a random oracle instead of H_1, and set $M_s\overrightarrow{p} = (H_1(ID_i'), \dots, H_1(ID_n'))^T$. In addition, the random seed r is sampled uniformly and the random function $row\,(ID, r)$ is defined by XoPaXoS. XoPaXoS implements a distribution over the values assigned to each position of the vector $\overrightarrow{p}$, whereby either a uniform random value is assigned or the value is the sum of the previous positions and a given offset $z_i - F_k(ID_i)$. This protocol ensures an equitable spread of possible values for each $\overrightarrow{p}$. Another security issue is that the malicious Server can play tricks like defining the OPRF function and running multiple hashing experiments to find collisions. In the malicious setting, the randomness of the OPRF function is further complicated by the fact that the random oracle is set to contain a random z negotiated by both parties. In addition, the Client can also check whether $c^s \neq H^F\,(w^s)$,where z^s is sent by the Server. Compared to PaXoS, XoPaXoS has the following advantages under the malicious model:

- For XoPaXoS, the position of the 0 value of $\overrightarrow{p}_i$ is replaced by a random value, and then solved by Gaussian elimination method to replace the fixed value generated by PaxoS. It is easy to verify that $\overrightarrow{p}$ values have an ideal uniform distribution.
- To solve this problem in XoPaXoS, tree vertices are also populated with random values. Subsequently, the remainder of the distribution $\overrightarrow{p_i}{:=}\mathrm{ID_i} \oplus \mathrm{p_j} \oplus \dots$ is generated, where each value of $\overrightarrow{p_i}$ corresponds to a value of $\mathrm{ID_i}$.

Theorem 4 (Client Privacy). *The PSKPIR protocol in malicious model can protect Client'privacy. The Client Privacy is based on the F_{OPRF}, $F_{PSI}^{payload}$-hybrid model with a random oracle, corresponding to Definition 4.*

Proof. This is different from step 3 in a semi-honest environment. We set $M\overrightarrow{p} = (H_1(ID_1), H_1(ID_2), \dots, H_1(ID_\mathrm{n}))^T$ to prevent corruption Server from solving linear systems, invocations are independent and break the linear correlation. Therefore, the probability that corruption Server can guess $\{ID_i | 1 \le i \le \mathrm{n}\}$ is negligible. Moreover, since r is uniformly distributed and mapped to binary vectors with keywords, the mapping of each keyword is linear and independent. If any $v^{\mathcal{H}}(ID_i, r)$ query has been performed before, it will be aborted. Therefore, the mapping is also indistinguishable from different keywords in PSI-Payload, at least with probability $1/n + k^{-c}$ and contradicts Client privacy.

Theorem 5 (Server/data Privacy). *In the F_{OPRF}, F_{OPPRF}, and $F_{PSI}^{payload}$-hybrid model with a random oracle, our protocol in the malicious model is server privacy by definition 5.*

Proof. We pay more attention to the probability of corruption of a probabilistic polynomial-time algorithm. Client can guess $\{(ID_i', D_i) | ID_i' \notin X\}$. This is different from step 6 in a semi-honest environment. We use the Decode algorithm proposed by XoPaXoS and select a random value of z in the malicious model. As a result of these modifications, the Decode algorithm is transformed into a linear function for $\overrightarrow{p}$, thus altering its original nonlinear form. The Decode exhibits the feature that $Decode(\overrightarrow{P}\Delta, ID_i) \neq H_1(ID_{i+1})$ is valid on all other positions, resulting in an incongruity of outputs, saving for a negligible probability. In addition, it is distributed uniformly for the corrupted Client and we use a random oracle to resolve the linear correlation to end. Server/data privacy is met by malicious OPRF construction. Futhermore, the Server also establishes the OPPRF using the XoPaXoS, ensuring that each position in the vector is either assigned a uniformly random value or computed as the sum of the previous positions and offsets $H_1(ID_i') - F'(ID_i')$. XoPaXoS-based cuckoo-graph is uniformly sampled from all possible (n, m)-cuckoo graphs, resulting in a collision probability of $2^{-(\lambda+d)}$. Since both λ and d are constant polynomial values and the probability is considered negligible, this approach is consistent with Server/data privacy.

7 Performance Evaluation

7.1 Experimental Details and Results

We develop PSKPIR on the basis of volePSI [27], and the implementation of related work depends on libPSI library. All our experiments were implemented on a benchmark machine with Intel Core i5 2.4 GHz, 16 GB RAM, 8 physical cores (all implementations are single-threaded). The network types were simulated using the Linux command t_c. Specifically, a LAN setting with 0.02 ms round-trip latency and 1 Gbps network bandwidth was used, as well as a WAN setting with a simulated 80 ms round-trip latency and different network bandwidths including 100 Mbps, 10 Mbps, and 1 Mbps. The variables n_1 and n_2 held the same number of input elements in $n_1, n_2 \in \{2^{12}, 2^{16}, 2^{20}\}$. The best protocol within a setting is marked in gray.

PSI with Payload. Here, we compare our semi-honest PSI with the work of [16,20,22] to demonstrate the benefits of our PSI in trade-off of communication costs and runtime when building PSI-Payload. The results of our evaluation in the semi-honest setting are shown in Table 1. The protocol proposed by Meadows et al. [20] exhibits a relatively minimal communication overhead, yet its execution speed is notably slow, being approximately 14× slower than our work. This is because protocol of Meadows et al. [20] requires computationally expensive public key operations, resulting in an accelerated increase in runtime as the number of sets increases. However, it has fewer communication instances compared to OT-based protocols [16,22]. Conversely, the protocol introduced by

Table 1: Comparison of Our PSI to Previous Works in the Semi-honest Setting.

N	Protocol	Communication (MB)			Running time (ms)			
		Client	Server	Total	LAN	100Mbps	10Mbps	1Mbps
2^{12}	Meadows [20]	0.16	0.13	0.29	3,438	28,686	47,641	998,114
	Kolesnikov [16]	0.11	0.32	0.43	150	460	1,860	3,551
	Pinkas [22]	0.66	0.08	0.74	160	489	6,235	15,920
	Our PSI	0.3	0.06	0.36	42	200	1,650	14,550
2^{16}	MEAMeadows [20]	2.63	2.06	4.69	45,369	466,606	7,743,818	9,761,826
	Kolesnikov [16]	1.7	5.2	6.9	412	1,596	4,558	57,565
	Pinkas [22]	9.13	1.06	10.19	235	1,682	11,860	12,280
	Our PSI	1.554	3.345	4.899	172	451	3,277	31,180
2^{20}	Meadows [20]	33.96	37.06	71.02	657,509	5,007,681	15,070,325	24,212,419
	Kolesnikov [16]	50.02	84.28	134.30	1,891	9,948	77,264	961,084
	Pinkas [22]	140.12	20.20	160.32	5,378	24,090	195,600	4,052,400
	Our PSI	2.03	50.03	52.06	4,398	8,496	48,690	449,700

Kolesnikov [16] performed faster speed, making it more suitable for deployment in low bandwidth environments, surpassing all other protocols in LAN setting. This is due to the fact that their can be preprocessed through a cuckoo hash, greatly reducing its runtime. However, if the input sets are larger and the bandwidth is reduced, the communication costs and runtime of protocol may increase significantly due to the need to send the OPRF value multiple times. Finally, Pinkas et al. [22] protocol avoids expensive computational overhead, although at the cost of enhanced communication compared to Meadows et al. [20] protocol. For medium input sizes and bandwidths, the protocol of Pinkas et al. [22] sometimes outperforms the protocol proposed by Kolesnikov et al. [16] protocol, thanks to its use of Paxos compression and data transfer. Our PSI utilizes not only the efficient running speed of Pinkas et al. [22] protocol but also leverages VOLE-based sublinear communication consumption to address the limitations of OT relied upon by Pinkas et al. [22] protocol. As a result, our approach reduces communication costs by 2×, and it is more applicable for large data sets.

In Table 2, we present the state of the art in the malicious setting, as shown by De et al. [6], Pinkas et al. [22], Rindal et al. [25], and Rindal(SM) et al. [26]. Our PSI demonstrates significantly faster performance compared to the blind-RSA protocol proposed by De et al. [6]. This improvement can be attributed to the execution of multiple exponentiation operations. In the protocol introduced by Rindal et al. [25], a substantial portion of the communication takes place during the OT extension phase, resulting in a total communication cost approximately 10× than that of our PSI. Furthermore, our PSI outperforms the LAN setting protocol proposed by Rindal(SM) et al. [26] for $\sigma = 64$. Although the performance is weakly dependent on σ, it is worth noting that our PSI remains competitive with their previously reported fastest protocol while eliminating the random-oracle assumption. However, in the WAN setting, the performance of the Rindal(SM) et al. [26] is limited by OT communication overhead, resulting in slightly slower performance than their improved protocol, Rindal et al. [25]. While our implementation consistently outperforms the protocol of Pinkas et

Table 2: Comparison of Our PSI to Previous Works in the Malicious Setting.

N	Protocol	Communication (MB)			Running time (ms)			
		Client	Server	Total	LAN	100Mbps	10Mbps	1Mbps
2^{12}	De [6]	0.32	0.51	0.83	22,400	53,200	131,004	818,500
	Rindal [25]	4.80	4.28	9.08	900.0	9,600	13,200	452,000
	Rindal(SM) [26]	179	179.53	358.53	752.9	5,680	78,940	156,800
	Pinkas [22]	1.6	3.1	4.7	135	225	1,895	22,450
	Our PSI	0.26	0.51	0.77	56	120.7	1,699	17,890
2^{16}	De [6]	5.13	8.20	13.33	365,000	567,000	793,000	1,275,054
	Rindal [25]	82.15	72.03	154.18	9,700	76,000	129,790	2,451,654
	Rindal(SM) [26]	1,521.03	1,515.97	3,037.01	6,172.8	565,000	1,563,400	89,124,000
	Pinkas [22]	12.62	2.097	14.717	651	1,808	13,130	125,500
	Our PSI	4.5	4.7	9.2	454	989	42,360	465,960
2^{20}	De [6]	82.13	131.13	213.26	305,0000	4,634,000	8,721,000	21,101,854
	Rindal [25]	693.53	608.03	1,301.56	127,000	1,480,000	16,163,245	18,963,210
	Rindal(SM) [26]	11,182	11,001	22,183	13,4000	7,654,000	26,663,000	455,889,921
	Pinkas [22]	63.55	254	317.55	6,119	10,348	540,900	4,950,000
	Our PSI	56	59	115	3,489	7,856	20,719	780,075

al. [22] in scenarios with reduced bandwidth, the latter demonstrates superior performance within a LAN. For 2^{20} elements, Pinkas et al. [22] protocol employs almost 11× less communication than Rindal et al. [25] protocol. This difference can be attributed to a more optimized implementation in the protocol of Pinkas et al. [22]. In terms of runtime, Pinkas et al. [22] outperforms Rindal et al. [25] protocol by a factor of approximately 10.5 in the LAN and by a factor of 20 in settings with a bandwidth of 1 Mbps. This disparity is due to the $\log n$ factor in computation that the protocol of Rindal et al. [25] requires. It is likely that the greater enhancement observed in the WAN configuration is due to the more pronounced impact of communication improvements on WAN performance.

Table 3: Comparison of PSKPIR to Previous PPIT Works.

N	Protocol	Communication (MB)			Running time (ms)			
		Client	Server	Total	LAN	100Mbps	10Mbps	1Mbps
2^{12}	RSA-PPIT	2.2	27.4	29.6	210	3,150	37,800	429,200
	Schnorr-PPIT	0.6	16.43	17.03	204	2,048	24,576	107,200
	IBE-PPIT	0.7	6.6	7.3	568	6,826	85,325	928,840
	Our	0.35	1.1	1.45	106	170.2	2,382.8	2,553
	Our (mal)	0.48	2.0	2.48	110	175.9	1,875.1	2,568
2^{16}	RSA-PPIT	128	346	474	4,033	50,412	645,273	40,066,268
	Schnorr-PPIT	30.3	80.6	112.9	2,100	32,768	4,65,305	14,20,403
	IBE-PPIT	15.57	58.06	73.63	60,014	109,226	1,551,009	7,271,827
	Our	1.60	16.62	18.22	189	2,161.1	30,688	478,726
	Our (mal)	4.83	21.06	25.89	256	2,493.4	38,897	707,926
2^{20}	RSA-PPIT	1,714	5,890	7,604	12,483	174,762	2,184,533	22,282,240
	Schnorr-PPIT	765	1,053	1,818	43,690	524,288	4,928,307	13,861,934
	IBE-PPIT	500	201	701	116,508	1,017,916	16,078,159	94,861,139
	Our	22	226	248	3,610	41,511.6	423,418	6,012,540
	Our (mal)	56	331	387	4,082	41,980.3	507,962	5,841,563

PPIT. In this analysis, we compare PSKPIR implementations against the works of RSA-PPIT, Schnorr-PPIT and linear asymptotic overhead protocol IBE-PPIT, respectively, and the results of our evaluation are shown in Table 3. Schnorr-PPIT and RSA-PPIT are two-round interactive protocols that utilize power multiplication several times during computation, resulting in a quadratic computation overhead. As the size of the set grows, our OT-based protocol features fast computing, with performance that is 20× and 10× better than RSA-PPIT and Schnorr-PPIT protocols in LAN settings, and 12× and 10× better in WAN settings, respectively. IBE-PPIT completes the protocol within a single round of interaction, with the calculation cost mainly consumed by bilinear mapping, and the communication cost being linear. However, its protocol transmission data primarily relies on asymmetric encryption. Compared with our protocol in a large data set, the running time of IBE-PPIT is 10× longer, and the communication cost is 3× higher in LAN settings.

Acknowledgements We gratefully acknowledge the support by National Natural Science Foundation of China (Grant no.U21A20463). This work benefited from the use of the VolePSI and libPSI Facility at the Visa-Research.

References

1. Ahmad, I., Agrawal, D., Abbadi, A.E., Gupta, T.: Pantheon: Private retrieval from public key-value store. Proceedings of the VLDB Endowment **16**(4), 643–656 (2022). https://doi.org/10.14778/3574245.3574251
2. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith, W.E.: Public key encryption that allows pir queries. In: Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27. pp. 50–67. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_4
3. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector ole. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 896–912 (2018). https://doi.org/10.1145/3243734.3243868
4. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. Journal of the ACM (JACM) **45**(6), 965–981 (1998)
5. De Cristofaro, E., Jarecki, S., Kim, J., Tsudik, G.: Privacy-preserving policy-based information transfer. In: Privacy Enhancing Technologies: 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings 9. pp. 164–184. Springer (2009)
6. De Cristofaro, E., Kim, J., Tsudik, G.: Linear-complexity private set intersection protocols secure in malicious model. In: Asiacrypt. vol. 6477, pp. 213–231. Springer (2010). https://doi.org/10.10007/1234567890
7. Demmler, D., Rindal, P., Rosulek, M., Trieu, N.: Pir-psi: scaling private contact discovery. Cryptology ePrint Archive (2018)
8. Di Crescenzo, G., Malkin, T., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 122–138. Springer (2000). https://doi.org/10.10007/1234567890

9. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 789–800 (2013). https://doi.org/10.1145/2508859.2516701
10. Freedman, M.J., Hazay, C., Nissim, K., Pinkas, B.: Efficient set intersection with simulation-based security. Journal of Cryptology **29**(1), 115–155 (2016)
11. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23. pp. 1–19. Springer (2004). https://doi.org/10.1007/978-3-540-24676-3_1
12. Gertner, Y., Ishai, Y., Kushilevitz, E.: Protecting data privacy in private information retrieval schemes. Journal of computer and system sciences **60**(3), p.592–629 (2000)
13. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 151–160 (1998)
14. Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Outsourced symmetric private information retrieval. In: Proceedings of the 2013 ACM SIGSAC conference on Computer&communications security. pp. 875–888 (2013)
15. Jarecki, S., Liu, X.: Fast secure computation of set intersection. In: Security and Cryptography for Networks: 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings 7. pp. 418–435. Springer (2010)
16. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious prf with applications to private set intersection. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 818–829 (2016)
17. 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 (2017). https://doi.org/10.1145/3133956.3134065
18. Lin, C., Liu, Z., Malkin, T.: Xspir: Efficient symmetrically private information retrieval from ring-lwe. In: Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I. pp. 217–236. Springer (2022)
19. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Information Security: 8th International Conference, ISC 2005, Singapore, September 20-23, 2005. Proceedings 8. pp. 314–328. Springer (2005)
20. Meadows, C.: 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. pp. 134–134. IEEE (1986). https://doi.org/10.1109/SP.1986.10022
21. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Crypto. vol. 99, pp. 573–590. Springer (1999)
22. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Psi from paxos: fast, malicious private set intersection. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II. pp. 739–767. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_25

23. Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based psi with linear communication. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38. pp. 122–153. Springer (2019). https://doi.org/10.1007/978-3-030-17659-4_5
24. Rabin, M.O.: How to exchange secrets with oblivious transfer. Cryptology ePrint Archive (2005)
25. Rindal, P., Rosulek, M.: Improved private set intersection against malicious adversaries. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I. pp. 235–259. Springer (2017)
26. Rindal, P., Rosulek, M.: Malicious-secure private set intersection via dual execution. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1229–1242 (2017)
27. Rindal, P., Schoppmann, P.: Vole-psi: fast oprf and circuit-psi from vector-ole. In: Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II. pp. 901–930. Springer (2021). https://doi.org/10.1007/978-3-030-77886-6_31
28. Sun, H., Jafar, S.A.: The capacity of symmetric private information retrieval. IEEE Transactions on Information Theory **65**(1), 322–329 (2018). https://doi.org/10.1109/TIT.2018.2848977
29. Tajima, A., Sato, H., Yamana, H.: Outsourced private set intersection cardinality with fully homomorphic encryption. In: 2018 6th International Conference on Multimedia Computing and Systems (ICMCS). pp. 1–8. IEEE (2018)
30. Wang, X., Luo, T., Li, J.: An efficient fully homomorphic encryption scheme for private information retrieval in the cloud. International Journal of Pattern Recognition and Artificial Intelligence **34**(04), 2055008 (2020)
31. Wang, Z., Ulukus, S.: Symmetric private information retrieval with user-side common randomness. In: 2021 IEEE International Symposium on Information Theory (ISIT). pp. 2119–2124. IEEE (2021)
32. Wang, Z., Ulukus, S.: Digital blind box: Random symmetric private information retrieval. In: 2022 IEEE Information Theory Workshop (ITW). pp. 95–100. IEEE (2022)
33. Yu, M., Yang, K., Wei, L., Sun, J.: Practical private information retrieval supporting keyword search in the cloud. In: 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP). pp. 1–6. IEEE (2014)

A Security Proof in Malicious Mode

Theorem 6. *The PSKPIR protocol realizes the $F_{PSI}^{Payload}$ functionality against a malicious adversary in the random oracle, F_{OPRF} and F_{vole}-hybrid model.*

Proof. Consider a Malicious Server. The simulator plays the role of F_{vole} and generates the Server's transcript as follows:

1) When $\mathcal{A}$ sends $(start, sid)$ to the F_{vole}, the simulator waits for $\mathcal{A}$ to send $\Delta, \overrightarrow{B}$. Then simulator sends uniform r, $\overrightarrow{A} = \overrightarrow{p} + \overrightarrow{A'}$ to $\mathcal{A}$.

2) In the event that the query for $F'_{\overrightarrow{K}}(ID\mathrm{i}')$ is initiated by $\mathcal{A}$, and given that $Decode(\overrightarrow{K}, ID'_i) - \Delta H_1(ID'_i) + z$ has not been queried beforehand, the simulator transmits $(start, sid, ID'_i)$ to F_{oprf} and establishes $F'_{\overrightarrow{K}}(ID'_{\mathrm{i}})$ as the response. Then the simulator sends Y to $F^{payload}_{PSI}$ and receives $\overrightarrow{p'}$ to $\mathcal{A}$.
3) The $\mathcal{A}$ sends Y', the simulator samples $Y^* = \{ID'_i | F'_{\overrightarrow{k}}(ID'_{\mathrm{i}}) = F'_{\overrightarrow{k}}(ID_i), ID'_{\mathrm{i}} \neq ID_i\}$ and computes $\widehat{Y'} = \{ID'_i | ID'_i \in Y^*, F'_{\overrightarrow{k}}(ID_i) \in Y'\}$. Then the simulator sends $\widehat{Y'}$ to $F^{Payload}_{PSI}$ and receives $\overrightarrow{p'}$ to $\mathcal{A}$.

To prove that this simulation is indistinguishable consider the following hybrids:

Expt0. The same as the real protocol except the simulation in this hybrid plays the role of F_{vole} .

Expt1. The previous experiment is now modified: The simulator sends r, $\overrightarrow{A} = \overrightarrow{P} + \overrightarrow{A'}$ to $\mathcal{A}$. Recall $\overrightarrow{A'}$ that is distributed uniformly in the view of $\mathcal{A}$, this $\overrightarrow{A}$ has an identical distribution. Besides, When $\mathcal{A}$ queries after samples r, if $v^{\mathcal{H}}(r, \cdot)$ has previously been made, it aborts. Due to the distributed uniformly sampling of r, making it impossible to differentiate distinguishable from the previous. Therefore, computational indistinguishability of $\mathcal{A}$'s view in **Expt0** and **Expt1** follows.

Expt2. We have modified the previous experiment by having the simulator call Encode and terminate if Encode fails. Since none of the queries $F'_{\overrightarrow{K}}(ID')$ and $H_1(ID')$ have been made before, ID' is uniformly sampled and the probability of termination is negligible. Thus, this hybrid is statistically indistinguishable from the previous one.

Expt3. This study modified the previous experiment by implementing protocol F_{OPRF} with $\mathcal{A}$. Whenever $\mathcal{A}$ requests $F'_{\overrightarrow{\mathrm{K}}}(ID'_{\mathrm{i}})$, the simulator sends ID'_i to F_{oprf} and instructs $F'_{\overrightarrow{\mathrm{K}}}(ID'_{\mathrm{i}})$ to the response if $Decode(\overrightarrow{K}, ID'_i) - \Delta H_1(ID'_i) + z$ has not been requested previously. Otherwise, $F'_{\overrightarrow{\mathrm{K}}}(ID'_{\mathrm{i}})$ responds normally. It is worth noting that z is uniformly distributed before transmission. As a result, any particular $Decode(\overrightarrow{K}, ID'_i) - \Delta H_1(ID'_i) + z$ follows a similar distribution, and it is improbable that $\mathcal{A}$ has previously requested $F'_{\overrightarrow{\mathrm{K}}}(ID'_{\mathrm{i}})$. Consequently, we can deduce that this hybrid is indistinguishable from the simulation.

Expt4. Now consider some collision $F'_{\overrightarrow{K}}(ID'_{\mathrm{i}}) = F'_{\overrightarrow{K}}(ID_i)$. Observations suggest that the simulator would only need to extract the values of ID'_{i} and ID_i if there is a significant likelihood of one of these values being present in X. Therefore, consider the probability of $F'_{\overrightarrow{K}}(ID'_{\mathrm{i}}) = F'_{\overrightarrow{K}}(ID_i)$ for some $ID_i \in X$. Since $|X| = |Y|$, the probability of the sender finding such a collision is negligible. Moreover, the simulator calls the $H(x)$ hash function, which is also used to eliminate collisions.

Therefore,computational indistinguishability of $\mathcal{A}$'s view in **Expt4** and **Expt3** follows. Since **Expt4** corresponds to a real-world execution of the protocol, this completes the proof.

Proof. Consider a Malicious Client $\mathcal{A}$. The simulator plays the role of F_{vole} ,and interacts with the Client as follows:

1) When $\mathcal{A}$ sends $\overrightarrow{A'}, \overrightarrow{C}$ to the F_{vole}. Upon transmitting of $r, \overrightarrow{A}$ by $\mathcal{A}$, the simulator calculates $\overrightarrow{p}$ and performs a check on $< \overrightarrow{P}, r, ID'_i >= H(ID_i)$ for each of the preceding $H\left(ID_i\right)$ queries made by $\mathcal{A}$. If the check is successful, it adds ID_i to the set X.
2) The simulator random sample $z \leftarrow \{0,1\}^k$ and proceeds to program: $F'(ID_i) := \{H(Decode(\overrightarrow{C}, ID_i) + z, ID_i), 1 \leq i \leq n\}$ for each $ID_i \in X$. Then it forwards X to $F_{PSI}^{payload}$ and receives $\widetilde{X} = \{(ID_i, D_i) | ID_i \in ID_i = ID'_i\}$ in response.
3) The simulator random sample $\overrightarrow{p}$ such that $F'_{\overrightarrow{K}}(ID_i) = \{H(Decode(\overrightarrow{K}, ID_i, r) - \Delta H_1(ID_i) + z, ID_i), ID_i \notin X\}$.
4) The simulator computes Y' as containing all $\{H(F(ID_i^*)) | ID_i^* \in ID \cap ID'\} \cup \{H(F(ID_i^*)) | ID_i^* \notin X\}$.The simulator sends Y'.

To prove that this simulation is indistinguishable consider the following hybrids:

Expt0. The same as the real protocol except the simulation in this hybrid plays the role of F_{vole}. The simulator waits for $\mathcal{A}$ to send $\overrightarrow{A'}, \overrightarrow{C}$. Whenever $\mathcal{A}$ queries $H\left(ID_i\right)$, the simulator checks whether $< \overrightarrow{P}, r, ID'_i >= H(ID_i)$. If the check is successful, it adds ID_i to the set X.

Expt1. Whenever $\mathcal{A}$ queries $H\left(ID_i\right)$, the simulator checks whether $< \overrightarrow{P}, r, ID'_i >= H(ID_i)$. If the check is successful, it adds ID_i to the set X. The simulator sends $(start, sid, X)$ to F_{oprf} and programs $X' := \{F_k(ID_i) | ID_i \in X\}$ to the response. Otherwise, X' responds normally.

Expt2. Upon sampling $z \leftarrow \{0,1\}^k$, the simulator checks if any $H(Decode(\overrightarrow{C}, ID_i) + z, ID_i)$ has been previously computed by $\mathcal{A}$ and aborts if this is the case. However, since z is just sampled, all $Decode(\overrightarrow{C}, ID_i) + z$ are uniformly distributed, making the probability of abort negligible. The simulation programs $F'(ID_i) := H(Decode(\overrightarrow{C}, ID_i) + z, ID_i)$ for all $ID_i \in X$. Given the uniformity of the $F'(x)$ function, the addition of programming H does not result in any changes to the distribution. Therefore, computational indistinguishability of $\mathcal{A}$'s view in **Expt1** and **Expt2** follows.

Expt3. Now we change the previous experiment:
The simulator aborts if $\mathcal{A}$ ever makes a $F'_{\overrightarrow{k}}(ID_i)$ query such that $F'_{\overrightarrow{K}}(ID_i) = \{H(Decode(\overrightarrow{K}, ID_i, r) - \Delta H_1(ID_i) + z, ID_i), ID_i \notin X\}$. However, through calculation and observation: $Decode(\overrightarrow{K}, ID_i, r) - \Delta H(ID_i) = \Delta(< \overrightarrow{p}, v^H(ID_i, r) > -H(ID_i)) + < \overrightarrow{C}, v^H(ID_i, r) >$. For $X = \{ID_1, \ldots, ID_n\}$ and $r \in \{0,1\}^k$, the $\mathcal{A}$ can trivially construct the unique $\overrightarrow{p}$ such that $< \overrightarrow{p}, v^H(ID_i, r) >= H(ID_i)$ for all $ID_i \in X$. But for $ID_i^* \notin X$, Since $H(x)$ is a hash function and all $< \overrightarrow{p}, v^H(ID_i^*, r) >$ are fixed, the probability of $< \overrightarrow{p}, v^H(ID_i^*, r) >= H(ID_i^*)$ is negligible. Therefore, this hybrid is statistically indistinguishable from the previous.

The $\overrightarrow{p}$ in previous experiment is modified and the simulator samples $\overrightarrow{p}$ by XoPaXoS:

1) Let random values be assigned the redundant p_i positions: $p_i \leftarrow \{0,1\}^{\mathrm{m+d+\lambda}}$.
2) For the remaining positions of $\overrightarrow{p}$ in R, define: $\mathrm{ID}'_i = ID_i - < M_i \overrightarrow{p_i} >$, otherwise p_i is assigned zero. As a fully determined system, there exists a unique solution.
3) Each tree in $\mathbb{G}$ assigns a uniform value to a singular node i.
4) In summary, the residual assignments can be represented as $p_i = v_k + p_j + \ldots$, with the stipulation that every assignment comprises an exclusive uniformly distributed v_k value, thereby ensuring the uniformity of p_i as intended.

It can be deduced from the security of XoPaXoS that the distribution of the output vector $\overrightarrow{p}$ in **Expt4** is computationally indistinguishable from the output distribution in the ideal-world scenario. As **Expt4** reflects the actual implementation of the protocol in the real-world setting, this completes the proof.