ASKPIR: Authorized Symmetric Keyword Privacy Information Retrieval Protocol Based on DID *

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 stronger PIR protocol that ensures both client and server privacy. In many cases, the client needs authorization from the data subject before querying data. However, this also means that the server can learn the identity of the data subject. To solve such problems, we propose a new SPIR primitive, called authorized symmetric keyword information retrieval protocol (ASKPIR). Specifically, we designed an efficient DID identification algorithm based on the Pedersen Commitment, which is used to solve the identity management and privacy problems of data subject when data is shared by multiple parties in a distributed environment. Then, we present a novel authorization algorithm combining NIZK proof and DID, which can preserve client privacy. Finally, to improve the efficiency of client retrieval, our protocol constructs PSI-Payload with mqRPMT and OTE so as to support batch keyword searches. In addition, we provide a formal security analysis for the anonymity and unforgeability of the protocol and demonstrate that ASKPIR can achieve malicious security under the UC framework. Theoretical analysis and experimental results show that the ASKPIR protocol is more efficient than other related works and solves the problem of incompatibility between data subject authorization and client privacy.

Keywords: Symmetric private information retrieval(SPIR) · Authorization · Decentralized identifier(DID) · Pedersen commitment · Non-interactive zero knowledge proof (NIZK) proof · Private set intersection with payload (PSI-Payload) · Oblivious transfer extension (OTE)· Multi-query reverse private membership test (mqRPMT) · Universal composability (UC).

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

1 Introduction

The symmetric private information retrieval (SPIR) protocol [10,19,24] is an extension of the private information retrieval (PIR) protocol [1,4,8] and there are generally two players: a client and a server. The server has a database that contains some value data, and the client submits a query item to the server in order to retrieve a particular data, where the items can be index-based[13,20] or keyword-based queries [15,29,28]. In general, the client usually does not know the index of the data. In some query scenarios, the client usually submits a keyword to the server for search and obtains the corresponding data [28,17,18], which is also the technology we focus on in this paper. The goal of SPIR is to maintain the privacy of both the client and the server [26,27]. That is, on the one hand, the client can retrieve data from the server without disclosing the specific keywords used in the query [25,29]. On the other hand, the server does not provide any additional information other than the data queried [19,24]. These properties have made SPIR one of the most promising candidate protocols for the PIR extension protocol [19,24,26,27]. SPIR has many practical applications. For instance, consider a financial scenario where two types of participants are involved: (1) a National Credit Center (NCC) containing the ID numbers and personal credits or loan records of consumer(C)s, (2) a bank (B) can query personal credit or loan records from the NCC using the consumer's ID number as a keyword. The B and the NCC can be thought of as the client and server, respectively, in a traditional SPIR setting. The requirements in the financial application are as follows:

Requirement 1 (Client privacy) *When the B queries NCC about C's personal credit or loan records, they do not want to reveal C's ID number to the server.*

Requirement 2 (Server privacy) *The B should not learn any personal information about the NCC, beyond the C's personal credit or loan records.*

However, from the C's perspective, B used his/her ID number to inquire about personal credit or loan records without authorization. In the same way, NCC also needs to obtain C's authorization to share personal credit or loan records with B. Therefore, we also need to introduce consumer C into the scenario, and C can be called a data subject who is the real owner of the personal credit or loan records that exist in the server. The data subject has the following requirements:

Requirement 3 (Data subject authorization) *The B should be authorized by C before using his/her ID number as a query keyword, and NCC should be able to verify whether B has a valid authorization from C.*

Limitation and Motivation The above example shows a typical scenario where common SPIR have two limitations: (1) Limitation 1: common SPIR cannot meet Requirement 1, 2 and 3 at the same time. According to the requirements 3, NCC needs to verify whether the ID number queried by B is authorized by

C, but the traditional SPIR protocol cannot meet the requirements 1. (2) Limitation 2: another limitation of SPIR is that batch keyword searching is also not implemented. In order to improve efficiency, B hopes to query the personal credit or loan records of multiple consumers (e.g. C_1,C_2, $\cdots$, C_n) at one time. Therefore, we need stronger SPIR to break through two limitations.

Currently, much research (cf., Table 1) has focused on the problem of authorization and batch keyword searching. De Cristofaro et al.[6] proposed a new primitive called privacy-preserving policy-based information transfer (PPIT). The protocol guarantees client and server privacy, and client input is controlled by well-defined access policies (such as authorization certificates). Then, in 2011, De Cristofaro et al.[7] proposed a privacy-preserving sharing of sensitive information protocol (PPSSI). The protocol utilized PSI to ensure the privacy of both the client and server. The client received data from the server, which requires authorization from the Certification Authority (CA). Goyal et al.[11] presented a novel framework for access control where only authorized clients can decrypt. Jarecki et al. [13] presented an Outsourced SPIR. The server is not able to learn about query keyword from the client while still being able to verify the validity of queries with the access policy. However, these related works in [22,7,13,11] do not allow data subjects to authorize, instead using the server or third-party to formulate access policies so that appropriate clients can retrieve the data. Fortunately, Song et al. [22] presented a protocol that allows for keyword search on encrypted data. The server can only process search queries on the stored ciphertext of data subjects if given proper authorization. Besides, it is unclear how the server can hide the identities of data subjects. layouni et al. [17] focused on Requirement 3 in SPIR. This protocol allowed data subjects to send anonymous credential [5] to clients for authorization while maintaining Requirement 1. Subsequently, they proposed a Multi-authorizer protocol [18] extend on protocol of [17]. However, the computational cost of both protocols is very expensive, and using a third-party CA can easily create a single point of failure. Additionally, both protocols are not suitable for data sharing among multiple parameter parties in a distributed environment.

Table 1: Comparison of the related surveys in the terms of Client privacy (T_1), Server privacy (T_2), PSI (T_3), Batch retrieval (T_4), Data subject authorization(T_5) and Decentralized certification authority (T_6) where ✓ indicates that topic was covered, × indicates that topic was not covered.

Reference	T_1	T_2	T_3	T_4	T_5	T_6
[6]	✓	✓	×	×	×	×
[7]	✓	✓	✓	✓	×	×
[11]	✓	×	×	✓	×	×
[13]	✓	✓	×	✓	×	×
[22]	×	✓	×	✓	×	×
[17]	✓	✓	×	×	Credential	×
[18]	✓	✓	×	×	Credential	×

Challenges and Goals: We summarize some technical challenges based on the two limitations and the problems described in some related works in Table 1.

- Challenge 1: In a distributed environment, when multiple participants share data, it is difficult to achieve unified identity management and ensure the identity privacy of each participant.
- Challenge 2: How to make the server complete authorization verification withoutviolating client privacy (i.e., Requirement 1) is difficult to achieve.
- Challenge 3: How to reduce the computational and communication costs of data subject authorization and client batch retrieval.

The goals of this paper is to introduce authorization in SPIR and address the issue of incompatibility with Requirement 1. In addition, we hope to establish a decentralized institution to manage the identity information of data subjects, so that the identity information of participants does not have to be disclosed when sharing data, and the anonymity of the identity of data subjects is maintained. Besides, we should optimize the computational and communication overhead of the protocol and enable client to receive authorized personal information in batches. Finally, we can formally construct a new primitive called ASKPIR, which is mainly used to solve the authorization problem in SPIR when multiple participants share data.

High-level Solution: In the ASKPIR setting, we have multiple players: a server, a client, and data subjects. The client sends query keywords (i.e., ID number of data subject) to the server for search. The server compares the stored keywords with the query keywords and responds with relevant data (i.e., personal information of data subject). The main contribution of ASKPIR is to ensure that the client has obtained the data subjects' authorization before the server processes the query keywords and that the server's verification of authorization does not reveal the data subjects' ID numbers.

The ASKPIR protocol we propose, combines DID (Decentralized identifier) with three cryptographic primitives: Pedersen Commitment, NIZK Proof and PSI-Payload. We designed a new type of DID identification method based on the Pedersen Commitment, which has the characteristics of global uniqueness, and is suitable for managing and hiding identities of data subjects in a distributed and multi-party data sharing environment. In principle, DID enable data subjects to manage their identities without the need to register with third parties. In addition, data subjects can participate in NIZK proof with DID and issue authorizations to the client. This enables client to prove to the server that they have the right to use ID numbers of data subjects without revealing any personally information about data subjects. This perfectly meets SPIR's security Requirement 1. Furthermore, during authorization, the generated Token can be used for pre-computation during client's keyword search phase, reducing computing consumption. In addition, we use mqRPMT and OTE to build PSI-Payload to add the function of batch keyword retrieval for customers, thus improving the query efficiency of the protocol. In general, the ASKPIR protocol can meet all requirements and solve all challenges mentioned above.

1.1 Our Contributions

Specifically, our contributions are summarized as follows:

- **Data Subject Authorization**. First, for the identity management and privacy of data subjects when sharing data with multiple parties, we can design an effective DID identification algorithm. The DID identification algorithms not only uniquely bind the identity of the data subject, guaranteeing the anonymity of the data subject without the need for third-party registration, but also allow servers to authenticate in a confidential and lightweight manner. For the privacy of the client, we propose a novel authorization algorithm combining Zero knowledge proof and DID. The data subject can authorize the client without exposing any information to the server during verification. For multi-data subject authorization, our protocol to build PSI-Payload, which facilitates the client to perform batch searches for authorized keywords and receive the corresponding payloads.
- **A New SPIR Primitive**. Secondly, based on the above ideas, we formally construct a new SPIR primitive through DID and PSI-Payload, ASKPIR, where DID is based on Pedersen Commitment and cooperates with NIZK proof to generate authorization. Our PSI-Payload is built on mqRPMT and OTE, where authorization pahse can realize offline pre-computation of client in mqRPMT and OTE allows client to calculate symmetric key locally by using keyword in the intersection, the resulting ASKPIR can achieve batch keyword search of client. Besides, we give a formal security model and propose new definitions of anonymity and unforgeability which address the threats in our scenarios. Furthermore, we demonstrate ASKPIR with security in the presence of malicious adversaries within the framework of Universal Composability (UC).
- **Effective Implementation**. Finally, the viability of the proposed ASKPIR protocol is evaluated through experimental results. Compared with related work, our computational and communication costs still advantages. Despite the introduction of data subject authorization, the additional cost of our build has a small impact on the protocol compared to the underlying building blocks. Source code is available at https://github.com/ZuodongWu2021/ASKPIR.

Outline: In the following section, we will review related work in Section 2 and then we outline the preliminary concepts and notations in the Section 3. In Sections 4, we give a system definition, algorithm and security model design of ASKPIR protocol. We describe our building blocks in Section 5. The main construction and security proof are displayed in Section 6. We evaluate the proposed protocol in terms of theoretical and practical performance in Sections 7 and 8, respectively. We conclude this work in Section 9. The detailed UC security proof of our protocol is described in Appendix A.

2 Related Work

Much research has gone into the problem of SPIR with authorization. For instance, Layouni [17] proposed an accredited SPIR (ASPIR) protocol based on

ElGamal homomorphic encryption and anonymous credential. Specifically, the data subject displays the public key along with a signed proof of knowledge to the server. Signing is performed on a challenge chosen by the server, which checks the validity of the credential by verifying the signature and credential also hides the identity of data subject. If the credential is valid, the server moves on to check the validity of the signed proof of knowledge. Moreover, for the credential security problem, this paper provides three solutions based on RSA and discrete logarithm. However, this protocol only allows the client to obtain authorization from a single user, and the client cannot retrieve data in batches. In order to prevent collusion attacks between server and CA, the protocol uses anonymous certificates for selective disclosure. However, anonymous credential can also increase the computational cost of the protocol, and there is also a single point of failure issue with CA. In 2008, Layouni et al. [18] further the ASPIR protocol of [17] to a protocol where each data can have multiple owners. This protocol uses a bilinear pairings-based signature scheme instead of anonymous credential and allows data subjects to encode, in the issued authorizations, any privacy policy they want to enforce on their data, including the client's identity, an expiry date etc. Although the constructions in [18] improves on the shortcomings of ASPIR [17], the two protocols still have common flaws: (1) The protocol uses a third-party CA and is not suitable for data sharing between multiple parties in a distributed environment. (2) Neither protocol considers whether the protocol can be safely executed in the presence of malicious client or server. Recently Jarecki et al. [13] proposed an outsourcing SPIR that supported authorization of data subject. The protocol allowed the server to provide search tokens to clients based on their queries and according to a given authorization policy. Besides, the protocol also showed that forging tokens or signatures is infeasible even by completely malicious clients. However, the protocol is more concerned with scenarios involving outsourcing, where multiple third-party clients retrieve data, but always need to be authorized by the data subject.

3 Preliminaries

3.1 Notations

We provide the pivotal concepts and description to be used in Table 2.

3.2 Cryptographic Assumptions

Let $Setup(1^k)$ be a PPT algorithm that on input a security parameter k, outputs description of an elliptic curve $G(\mathbb{F}_p)$ with base point G of of prime order $p = \Theta(2^k)$.

Assumption 1 (ECDLP Assumption) *Given the tuples $(G, a, a \cdot G)$, where $a \cdot G \in G(\mathbb{F}_p)$. We define the Elliptic Curve Discrete Logarithm (ECDLP) problem to hold if the advantage ε^{ECDLP} obtained by any polynomial-time adversary $\mathcal{A}$ is negligible under the security parameter k:*

$$\varepsilon^{ECDLP} := \Pr[A_{ECDLP}(G, a \cdot G) = a : a \in \mathbb{Z}_q] \leq negl(k) \tag{1}$$

Table 2: Notations and description.

Symbol	Definition
Z	The set of integer numbers
$[n]$	The set $1, \ldots, n$.
n_1	The input set of client.
n_2	The input set of server.
k	System security parameter
λ	Statistical security parameter.
M	The number of queries that F can ask to the random oracle.
F_1	A probabilistic polynomial time Turing machine.
sid	The identity of the client or the server.
$\mathbb{F}_p$	A field of prime numbers of order p.
l_1	The length of keyword.
l	The length of the data subject's personal data (Payload).
$\varOmega$	The Bloom filter.
F	Commutative weak PRF $F : K \times D \rightarrow D$, where K is the key space, D is domain.
$G(\mathbb{F}_p)$	An elliptic curve with base point G and order N, where p is a large prime number.
H_3	A random oracles, where $H_3 : [n_2] \cdot \{0,1\}^k \rightarrow \{0,1\}^l$.

Assumption 2 (ECDH Assumption) *Given the tuples $(G, a\cdot G, b\cdot G)$, where $a\cdot G, b\cdot G \in G(\mathbb{F}_p)$. We define the Elliptic Curve Diffie-Hellman (ECDH) problem to hold if the advantage ε^{ECDH} obtained by any polynomial-time adversary $\mathcal{A}$ is negligible under the security parameter k:*

$$\varepsilon^{ECDLP} := \Pr[A_{ECDH}(G, a\cdot G, b\cdot G) = (h, h^{ab}) \in G : h \in G(\mathbb{F}_p), a, b \in \mathbb{Z}_q] \leq negl(k) \tag{2}$$

3.3 The Forking Lemma

Theorem 1 (The Forking Lemma). *Let $(\eth, \varSigma, V)$ represent a digital signature scheme with a security parameter k. We use the symbol M to denote the total number of queries that F_1 is allowed to make to the random oracle. We assume that F_1 is capable of producing a valid signature $(m, \sigma_1, h, \sigma_2)$ within a specific time constraint of T^*, with a probability $\varepsilon \geq 7M/2^k$. By utilizing another machine, which has control over F_1, it is possible to generate two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$. The expected time required for this process is given by $T'^* \leq 84480T^*M/\varepsilon$.*

3.4 Decentralized Identifier (DID)

DID [9] is an identifier composed of a string that represents a digital identity which is a decentralized and verifiable identifier. DID does not require third-party registration, and data subjects can register different DIDs at different institutions. Data subjects can independently complete the registration, parsing, updating or revoking operations of DID, and can achieve global uniqueness without the need for a CA. Each DID is stored in the corresponding DID document, and any user can query the DID document on the Blockchain. Here is an example DID:

$$did : ASPIR : jfijf3reiEojeoureW$$

where did is the DID Scheme (similar to http,https,ftp and other protocols in the URL); $ASPIR$ is the DID Method Identifier (usually the name of the DID method); $jfijf3reiEojeoureW$ is the specific identifier in the DID method: it is unique in the entire DID method namespace. Its advantages are as follows:

- **Decentralization:** Based on Blockchain, identity is not controlled by a single centralized authority.
- **Identification and Authentication:** The identity-related data is recorded on the Blockchain, and the process of authentication does not need to depend on the data subjects providing the identities.
- **Trusted Multi-party Data Sharing:** The identity of the data subject can be managed and used in a decentralized environment, and the data subject can share the identity and data through authorization.

3.5 Private Set Intersection with Payload (PSI-Payload)

PSI-Payload [14] is a modified version of PSI, where each keyword belonging to the input set of the server is linked with a corresponding value, referred to as a payload.

PSI-Payload Ideal Functionality. The PSI-Payload protocol uses intersection operations to facilitate the safe computation of payload, while also ensuring that both parties'privacy is maintained. The PSI-Payload functionality is presented in Figure 1.

PSI with payload Ideal Functionality $F_{PSI}^{payload}$

Parameters:
X: a set $X = \{x_i, \ldots, x_{n_1}\}$ with n_1 of data.
Y: a set $\mathrm{Y} = \{(\mathrm{y}_1,\tilde{\mathrm{y}}_1), \ldots, (\mathrm{y}_{n_2},\tilde{\mathrm{y}}_{n_2})\}$ with n_2 sets of data, where $\tilde{\mathrm{y}}_i$ is a payload.
Input: Input a message ($start$, sid) from the server and ($start$, sid) from the client.
Functionality: (No Corrupted): Wait for the client send a set of X and the server send a set of Y.
(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 $\widetilde{\mathrm{X}}$= $\{(\mathrm{x_i}, \tilde{\mathrm{y}})|\mathrm{x_i} \in X \cap Y\}$ to the client.

Fig. 1: Ideal functionality $F_{PSI}^{payload}$ of PSI-Payload.

4 System Definition and Algorithm Definition

4.1 System Definition

Through the analysis in the introduction 1 section, it can be seen that the combination of DID and PSI-Payload can effectively solve the identity identification and authorization problems in SPIR under distributed environments. Therefore, this paper introduces the data subject and designs an effective DID identification algorithm to protect the identity privacy of the data subject in a distributed environment; Combined with zero-knowledge proof, we proposed a new authorization algorithm to solve the identity privacy and authorization issues of SPIR; Finally, in order to achieve batch retrieval and multi-data subject authorization, we use PSI-Payload to build the SPIR protocol. The system model is shown in Figure 2. Specifically, the system model is divided into 5 pahses:

(1) **Registration Phase:** Multiple data subjects register the DID with system parameters into the smart contract of the Blockchain, generating DID, DID documents, and some service information. DID documents and service information can be stored in the Blockchain. The client and server query the DID and other information of the data subject in the Blockchain.

(2) **Authorization Phase:** Each data subject generates authorization proof with the help of zero-knowledge proof, and sends the proof and other authorization information to the client.

(3) **Verification Authorization Phase:** The server receives the proofs and corresponding retrieval information. The server verifies whether the client has been authorized by the data subjects. If the verification is successful, the server can continue execution, otherwise the protocol is aborted.

(4) **Keyword Search Phase:** The client and server execute the PSI protocol and the client uses the $Tokens$ of multiple data subjects to retrieve corresponding payloads in batches, the server performs keyword matching, and the client receives the matching result.

(5) **Payload Transmission Phase:** The client and server execute the OTE protocol, and the client receives the payload corresponding to a successful keyword match with the server.

4.2 Algorithm Design

Based on the above ideas, we formally constructed a new primitive, ASKPIR, through DID and PSI-Payload, where DID is based on Pederson Commitment and combined with NIZK proof to generate authorization. Our PSI-Payload is built on mqRPMT and OTE. An ASKPIR protocol includes the following five polynomial time algorithms:

- $Setup(k, \lambda) \rightarrow (pp, SK_c, sk_i, sk_s, PK_c)$: On inputting the security parameter k and λ, the algorithm outputs public parameter pp, public-private key pair (SK_c, PK_c) of the client, private key sk_i of the data subjects, private key sk_s of the server.

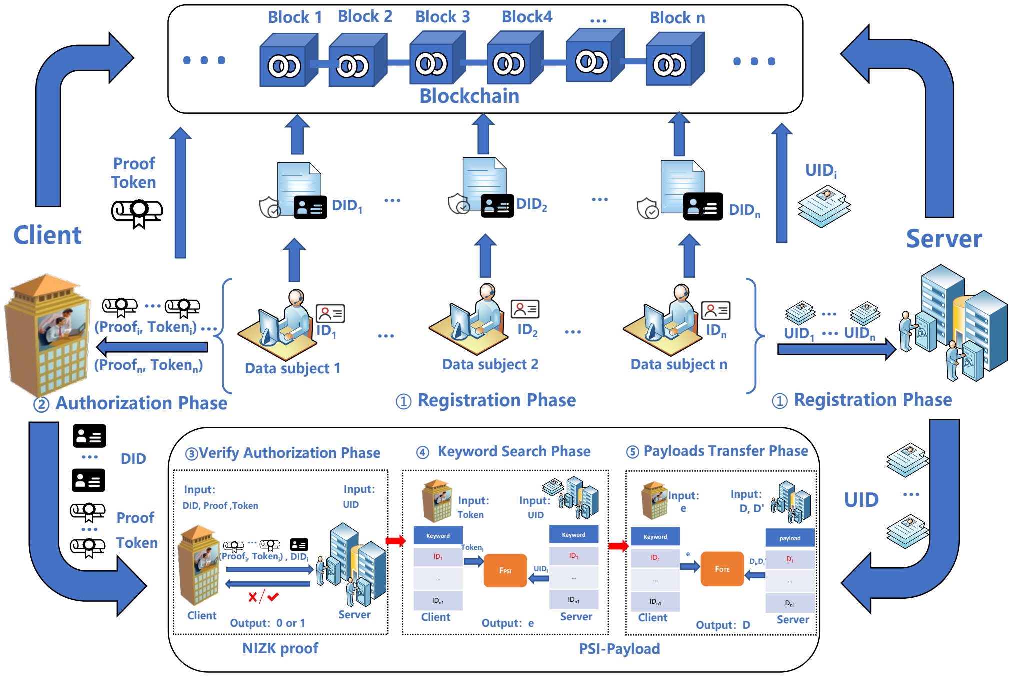

Fig. 2: System Model

- $RegDID(pp, sk_i, ID_i, \alpha_i) \rightarrow (DID_i, UID_i)$: The algorithm is performed by the data subject. On inputting pp, sk_i, ID_i, and a random integer $\alpha_i \in \mathbb{Z}_q$ to smart contracts in blockchain, the algorithm generates the DID_i and the corresponding DID document. The data subject sends UID_i to the server, where UID_i indicates computing rights to the data subject's personal data.
- $GenAut(DID_i, pp, PK_c) \rightarrow (Token_i, proof_i)$: This is an authorization algorithm executed by the data subject. On inputting public parameter pp, DID_i and PK_c, the algorithm outputs a $Token_i$ and $proof_i$.
- $VerAut(DID, PK_c, Token_i^*, proof_i^*) \rightarrow (0, 1)$: This algorithm is executed by the server. The input DID_i, PK_c, $Token_i^*$ and $proof_i^*$, the algorithm returns 0 or 1. 1 means that the verification is passed; 0 means that the verification fails and the next step is refused.
- $GenKS(Token, I, sk_s) \rightarrow e$: The algorithm is executed by the server and client. The client inputs a set $Token = (Token_1, \ldots, Token_{n_1})$, and the server inputs a set $I = \{(UID_i, D_1), \ldots, (UID_{n_2}, D_{n_2})\}$ and sk_s. The algorithm returns the indication bit e to the client, while the server learns nothing.
- $PayTrans(e, I) \rightarrow D_i$: The algorithm is executed by the server acts as a receiver and client as a sender with input e. The algorithm outputs the Client receives the Payloads corresponding to the keyword successfully matches in the server.

4.3 Security Model

In this section, we will describe the security model of the ASPIR protocol and provide definitions for two security attributes: Anonymity and Unforgeability.

Before formalizing the definition, we provided the following oracle through Challenger C to simulate adversary $\mathcal{A}$'s ability to attack the security of the protocol:

- $\mathcal{O}_{Reg}$: Register Oracle. Adversary $\mathcal{A}$ queries the oracle to obtain the DID_i and public-private key pair (sk_i, pk). The oracle is used to simulate that the $\mathcal{A}$ can register as a real data subject and legally obtain the correct DID and (sk_i, pk).
- $\mathcal{O}_{DID}$: Adversary $\mathcal{A}$ can obtain the corresponding DID_i, $Token_i$, and NIZK $proof_i$ on the Blockchain through the index. The oracle simulates that adversary $\mathcal{A}$ can obtain the authorization information of some honest data subjects by observing the Blockchain, and the DID Document also has a corresponding pk.

Anonymity. The Anonymity means that except the holder of the DID, no participant can obtain the real ID corresponding to the DID. We define the Anonymity through the following interactive experiment $Expt_{ASPIR}^{Anon}(k)$ between challenger C and adversary $\mathcal{A}$.

(1) *Setup*: C executes the public parameter generation algorithm $pp \leftarrow SysGen(1^{\mathrm{k}})$ and sends pp to $\mathcal{A}$;
(2) $Queries1$: $\mathcal{A}$ adaptively query $\mathcal{O}_{Reg}$ and $\mathcal{O}_{DID}$.
(3) *Challenge*: $\mathcal{A}$ sends (ID_0, ID_1) to C. C first selects a bit $\beta \xleftarrow{R} [0,1]$, then uses Pedersen Commitment to generate DID_β, and finally sends it to $\mathcal{A}$.
(4) $Queries2$: $\mathcal{A}$ continues to adaptively query $\mathcal{O}_{Reg}$ and $\mathcal{O}_{DID}$.
(5) *Guess*: $\mathcal{A}$ guesses bit $\beta' \in [0,1]$, if $\beta = \beta'$, then the experiment output is 1, $\mathcal{A}$ will win the experiment; Otherwise, the experiment outputs 0.

The advantage of adversary $\mathcal{A}$ in breaking the data subject anonymity is as follows:

$$Adv_{ASPIR}^{Anon}(k) = |\Pr\left[Expt_{ASPIR}^{Anon}(k) = 1\right] - \frac{1}{2}| = |\Pr\left[\beta = \beta'\right] - \frac{1}{2}| \quad (3)$$

Definition 1 (Anonymity). *For any PPT adversary $\mathcal{A}$, if the advantage of adversary $\mathcal{A}$ in breaking the anonymity is $Adv_{ASPIR}^{Anon}(k) \leq negl(k)$, then the protocol satisfies the anonymity.*

Unforgeability. The Authorization Unforgeability of authorization means that if the data subject's ID and sk_i are not leaked, the adversary $\mathcal{A}$ cannot forge correct authorization information (eg., DID_i, $Token_i$, NIZK $proof_i$). We define Unforgeability through the following interactive experiment $Expt_{ASPIR}^{Unfor}(k)$ between challenger C and adversary $\mathcal{A}$.

(1) *Setup*: C executes the public parameter generation algorithm $pp \leftarrow SysGen(1^{\mathrm{k}})$ and sends pp to $\mathcal{A}$;
(2) $Queries1$: $\mathcal{A}$ adaptively query $\mathcal{O}_{Reg}$ and $\mathcal{O}_{DID}$.

(3) *Forgery*: $\mathcal{A}$ forged NIZK proof: $\mathrm{proof_i}^* = A^{O_{\mathrm{Reg}} O_{DID}}(pp, DID_i^*, pk^*, Token_i^*)$
(4) *Queries*2: $\mathcal{A}$ continues to adaptively query $\mathcal{O}_{Reg}$ and $\mathcal{O}_{DID}$.
(5) *Judges*: C receives the NIZK Proof and outputs a judge($\mathrm{proof_i^*}$, pk, $\mathrm{Token_i^*}$, $\mathrm{DID_i^*}$) = true, the experiment outputs 1; otherwise, it outputs 0.

The advantage of adversary $\mathcal{A}$ in breaking the unforgeability of data subject as follows:

$$Adv_{ASPIR}^{Unfor}(k) = \Pr\left[Expt_{ASPIR}^{Unfor}(k) = 1\right] \tag{4}$$

Definition 2 (Unforgeability). *For any PPT adversary $\mathcal{A}$, if the adversary $\mathcal{A}$ has negligible advantage $Adv_{ASPIR}^{Unfor}(k)$ in breaking the unforgeability of data subject, that is, $Adv_{ASPIR}^{Anon}(k) \leq negl(k)$, then the protocol satisfies the unforgeability.*

5 Protocol Building Blocks

Below we will introduce the following building blocks to form a new primitive, ASKPIR.

5.1 Pedersen Commitment

Pedersen Commitment [21] is a commitment protocol that satisfies perfect hiding and computational binding. Below we recall the Pedersen commitment, as show in Figure 3:

Pedersen commitment $\mathit{\Pi}_{com}$

Parameters:
Input a security parameter k, output $pp = (G(\mathbb{F}_p), p, H, G)$, where $H \xleftarrow{R} G(\mathbb{F}_p)$.
Input:
Input $m \in \mathbb{Z}_p$ and $r \xleftarrow{R} \mathbb{Z}_p$, output $c = m \cdot G + r \cdot H$.
Outputs:
Output "1" if $c' = m' \cdot G + r' \cdot H$ and "0" otherwise.

Fig. 3: Protocol $\mathit{\Pi}_{com}$ which the Pedersen commitment functionality.

Definition 3 (Hiding). *A Pedersen commitment $Com(m, r) \to c$ should not disclose any information about its committed value m. Let $\mathcal{A}$ represent an adversary against hiding. We define the advantage of the following experiment:*

$$Adv(k) = \Pr\left[\beta' = \beta : \begin{array}{ll} pp \leftarrow Setup(k) & \\ (ID_0, ID_1) \leftarrow A_1(pp); & \\ \beta \leftarrow \{0,1\}, \beta \leftarrow \{0,1\} & c \leftarrow Com(ID_\beta; r) \\ \beta' \leftarrow A_2(c) & \end{array}\right] - \frac{1}{2} \tag{5}$$

The Pedersen Commitment is perfectly hiding under the discrete logarithm assumption and $Adv(k) = 0$ even for an unbounded adversary.

Definition 4 (Binding). *A Pedersen commitment c cannot be opened to two different messages. Let $\mathcal{A}$ denote an adversary against binding. We define the advantage of $\mathcal{A}$ through the following experiment:*

$$Adv(k) = \Pr\left[\begin{array}{c} m_0 \neq m_1 \wedge \\ c = Com(m_0, r_0) = Com(m_1, r_1) \end{array} : \begin{array}{c} pp \leftarrow Setup(k) \\ (c, m_0, m_1, r_1, r_0) \leftarrow \mathcal{A} \end{array}\right] \quad (6)$$

The Pedersen Commitment is perfectly binding under the discrete logarithm assumption and $Adv(k) = 0$ even for an unbounded adversary.

5.2 Multi-query reverse private membership test

The Reverse Private Membership Test (RPMT) [16] is a protocol where a client, with input X, interacts with a server that holds a set Y. After the interaction, the client learns only whether $x_i \in Y$, while the server learns nothing about the set X. RPMT typically focuses on the client querying for a single element. RPMT can repeat this process multiple times, so it is called multi-query RPMT (mqRPMT). In Figure 4 we formally define the ideal functionality for mqRPMT.

Ideal functionality F_{mqRPMT} for mqRPMT

Parameters:
The client specifies the number of queries n_1 and the server's set size n_2.
Input: The server inputs a set $\mathrm{Y} = (\mathrm{y}_1,...,\mathrm{y}_{\mathrm{n}_1})$, where $\mathrm{y}_\mathrm{i} \in \{0,1\}^{l_1}$. The client inputs a set $\mathrm{X} = (\mathrm{x}_1,...,\mathrm{x}_{\mathrm{n}_2})$, where $\mathrm{x}_\mathrm{i} \in \{0,1\}^{l_1}$.
Outputs: The client receives a vector $\overrightarrow{e} = (e_1, \ldots, e_{n_2}) \in \{0,1\}$ where $e_i = 1$ if $x_i \in Y$ and $e_i = 0$ otherwise. The server receives no information.

Fig. 4: Ideal functionality F_{mqRPMT}.

In Figure 5, we show how to build mqRPMT from Commutative Weak PRF to realize the mqRPMT functionality F_{mqRPMT}.

5.3 Oblivious transfer extension (OTE)

An Oblivious transfer extension (OTE) [2,12] protocol involves using a small number of base OTs to obtain many OTs by using cheap symmetric cryptographic operations. Hybrid encryption is a process that involves encrypting large messages using a symmetric key instead of RSA, which would be too expensive. To achieve this, a single RSA computation is performed to encrypt the symmetric key, which is then used to encrypt the long message using only symmetric operations. OTE can be achieved with remarkable efficiency, and the security parameter is independent of the number of OTs, allowing it to be as small as 128. In this work, we need the primitives of OTE to build symmetric keys for payload transmission, which can be achieved through Figure 6, and the ideal functionality is defined in Figure 7.

MqRPMT from cwPRF Protocol Π_{mqRPMT}

Parameters:

The size of the client's set is denoted by n_1, and the size of the server's set is denoted by n_2. A cwPRF denoted by $F : K \times D \to D$ and a hash function denoted by H :$\{0,1\}^* \to D$ are used.

Input: The server inputs a set $\mathrm{Y} = (\mathrm{y}_1,...,\mathrm{y}_{\mathrm{n}_1})$, where $\mathrm{y}_\mathrm{i} \in \{0,1\}^{l_1}$. The client inputs a set $\mathrm{X} = (\mathrm{x}_1,...,\mathrm{x}_{\mathrm{n}_2})$, where $\mathrm{x}_\mathrm{i} \in \{0,1\}^{l_1}$.

- The client random selects $k_1 \xleftarrow{R} K$, then it sends the value of $\{F_{k_1}(H(x_i))\}_{i\in[n_1]}$ to server.
- The server random selects $k_2 \xleftarrow{R} K$. It then computes and sends the value of $\{F_{k_2}(H(y_i))\}_{i\in[n_2]}$ to client. The server also computes the value of $\{F_{k_2}(F_{k_1}(H(x_i)))\}_{i\in[n_1]}$ and picks a random permutation $\pi \xleftarrow{R} [n_1]$. It then sends the value of $\{F_{k_2}(F_{k_1}(H(x_{\pi_{(i)}})))\}_{i\in[n_1]}$ to the server.

Outputs: The client computes the value of $\{F_{k_1}(F_{k_2}(H(y_i)))\}_{i\in[n_2]}$, and sets $\mathrm{e}_\mathrm{i}= 1$ iff $F_{k_1}(F_{k_2}(H(x_i))) = F_{k_2}(F_{k_1}(H(x_{\pi_{(i)}})))$.

Fig. 5: Protocol Π_{mqRPMT} which realizes the ideal mqRPMT functionality.

Oblivious Transfer Extension Ideal Functionality F_{OTE}

Parameters:

Number of OT instances n and the length of the message, denoted by l.

Input:

The server inputs $\{(m_{i,0}, m_{i,1})\}_{i\in n}$, where each $m_{i,b} \in \{0,1\}^l$. The client inputs a bit vector $(\mathrm{b}_1, \ldots, \mathrm{b}_\mathrm{n}) \in \{0,1\}^n$.

Outputs:

The server does not receive anything, while the client receives $\{m_{i,b}\}_{i\in[n]}$.

Fig. 6: Ideal functionality F_{OTE} of OTE.

Oblivious Transfer Extension Protocol Π_{OTE}

Parameters:
A random oracle $H_3 : [m] \times \{0,1\}^k \to \{0,1\}^l$, where $|m|$ is the number of columns of the matrix.
Input: The server inputs m pairs $(x_{j,0}, x_{j,1})_{j \in [1,m]}$ of l-bit strings. The client inputs m selection bits $\mathrm{r} = (\mathrm{r}_1, \ldots, \mathrm{r}_\mathrm{m})$.

- The server initializes a random vector $\in \{0,1\}^n$ and client a random $m \times k$ bit matrix T.
- The parties utilize the OT primitive, where the server acts as a receiver with input s, and the client acts as a sender with inputs $(t^i, r \oplus t^i)_{i \in [1,k]}$.
- Let Q be the $m \times k$ matrix of values received by the server. Then, the server sends $(y_{j,0}, y_{j,1})$ where $y_{j,0} = x_{j,0} \oplus H_3(j, q_j)$ and $y_{j,1} = x_{j,1} \oplus H_3(j, q_j \oplus s)$.

Outputs: The client outputs $z_j = y_{j,r_j} \oplus H_3(j, t_j)$.

Fig. 7: Protocol Π_{OTE} which realizes the OTE functionality F_{OTE}.

5.4 Schnorr protocol

In our work, Schnorr protocol [23] are the main building blocks for NIZK Proof. The protocol is a way of proving knowledge of the elliptic curve discrete logarithm (ECDH) with a low probability of cheating (knowledge error) which is equal to 2^{-k}. The protocol is also designed to ensure that an honest verifier cannot gain any additional information.

Schnorr protocol $\Pi_{Schnorr}$

Parameters:
Input a security parameter k, output $pp = (G(\mathbb{F}_p), p, H, G)$, where $H_1 \xleftarrow{R} G(\mathbb{F}_p)$.
Input:

- The prover selects a random integer r_α , computes a commitment $t = r_\alpha \cdot G$. The challenge is then computed as $c = H_1(G||y||m||t)$, where m is the message and $PK\{(\alpha) : y = \alpha \cdot G\}$.
- The prover computes the protocol response $s = \mathrm{r}_\alpha - \mathrm{c} \cdot \alpha$. The proof consists of the pair (s, c) and sends it to the verifier.
- The verifier performs the computation $\hat{t} = G \cdot s + y \cdot c$ and subsequently validates the equation $c = H_1(g||y||\hat{t})$ for consistency.

Outputs: "1" if the verifier accepts the protocol and "0" otherwise.

Fig. 8: Protocol $\Pi_{Schnorr}$ which the Schnorr signature functionality.

6 Main Construction and Security Analysis

This section introduces the detailed construction and security proof. The ASKPIR involves three players:

- **Data Subject(DU):** The actual owner of the payload and ID numbers in the server.
- **Client(C):** uses the DUs' ID numbers $X = \{ID_i, ID_i \in \{0,1\}^{l_1}, i \in [n_1]\}$ to retrieve the corresponding payload. However, the client requires the authorization of the DUs to use ID_i.
- **Server(S):** stores the set $Y = \Big\{(ID'_i, D_i), ID'_i \in \{0,1\}^{l_1}, D_i \in \{0,1\}^{l}, i \in [n_2]\Big\}$, where ID'_i denotes a DU's ID number and D_i represents the associated payload.

6.1 ASKPIR

The ASKPIR protocol consists of six algorithms and the detailed algorithm design is as follows:

Setup Algorithm:($\boldsymbol{Setup(k, \lambda) \rightarrow (pp, SK_c, PK_c, sk_i, sk_s)}$). It inputs security parameters k and λ, generates SK_c, PK_c, sk_i, and sk_s, and generates public parameters pp. The specific steps are as follows:

1) The system runs the setup algorithm to produce an elliptic curve denoted as $G(\mathbb{F}_p)$ with base point G. Suppose H is a point on the elliptic curve, $H_1 : \{0,1\}^* \longrightarrow \{0,1\}^{l'}$ and $H_2 : \{0,1\}^* \longrightarrow \{0,1\}^{l'}$ are two random oracles, where $l' \geq \lambda + \log_2{}^{n_1 n_2}$, and the output public parameter $pp = \{G(\mathbb{F}_p), G, H, H_1, H_2, N\}$. It also outputs a $sk_s \overset{R}{\longleftarrow} [N-1]$ for the S, a $sk = \{sk_i \overset{R}{\longleftarrow} [N-1] | i \in [n_1]\}$ for the DUs, and a public-private key pair (PK_c, SK_c) for the C, where $SK_c \overset{R}{\longleftarrow} [N-1]$ and $PK_c = sk_c \cdot G$.

RegDID Algorithm:($\boldsymbol{GenDID(pp, sk_i, ID_i, \alpha_i) \rightarrow DID_i, UID_i}$). The DU_i takes ID_i to register with the Blockchain smart contract and generates a DID_i and a DID document. The DID document contains the DID_i and can be stored on the Blockchain. The specific registration steps are as follows:

1) DU_i uses a hash algorithm to generate $hid_i = H_1(IDType \parallel ID_i) \cdot \alpha_i$, where α_i is secret blinding factor randomly chosen in $\mathbb{Z}_p$ and the $IDtype$ denotes the class of identifier that is present within a keyword.
2) Then DU_i uses hid_i and sk_i to generate the unique identifier of the DID_i Method-Specific Identifier through the Pedersen Commitment. The computation process is as follows:
$$Com_{HID_i} = sk_i \cdot H + hid_i \cdot G \tag{7}$$
3) The DU_i sets $DID = did : ASKPIR : Com_{hid}$ and the DID document is stored in the blockchain. In addition, the DU_i calculated $UID_i = hid_i \cdot G$, which represents the right that the S can use ID_i to do computations. These $UIDs$ are also stored on the blockchain.

GenAut Algorithm: $\boldsymbol{(GenAut(DID, pp, pk_c) \rightarrow (Token_i, proof_i))}$**.** If C wants to use DU_i's ID as the keywords to retrieve the corresponding payload, they must obtain authorization from DU_i. The DU_i authorization process is as follows:

1) The DU_i first computes a Token for payload query authorization where $Token_i = hid_i \cdot PK_c$. Then, DU_i sends DID_i, $Token_i$ to the C.
2) The DU_i generates a $proof$ from DID and $Token$. Given H, G, PK_c, DID_i, N, $Token_i$, the DU_i to prove to S:

$$PoK\left\{ sk_i, hid_i : \begin{array}{c} DID_i = Comt_{HID_i} = sk_i \cdot H + hid_i \cdot G \\ \wedge \\ Token_i = hid_i \cdot PK_c \end{array} \right\} \tag{8}$$

   The DU_i constructs NIZK proof based on the Schnorr protocol and the Fait-Shamir heuristic:
   - Select $r_{1,i}, r_{2,i} \xleftarrow{R} Z_q$ and compute $Q_{1,i} = r_{1,i} \cdot G + r_{2,i} \cdot H$;
   - Compute $Q_{2,i} = r_{1,i} \cdot PK_c$;
   - Compute $c_i = H_1 \left(G \,\|H\|\, DID_i \,\|N\|\, Token_i || PK_c || Q_{1,i} \,\|Q_{2,i}\|\right)$;
   - Compute $s_{1,i} = r_{1,i} - c_i \cdot hid_i$;
   - Compute $s_{2,i} = r_{2,i} - c_i \cdot sk_i$;
3) The DU sends a parameter of NIZK Proof: $proof_i = (c_i, s_{1,i}, s_{2,i})$ to the C.

   VerAut Algorithm:$\mathbf{(VerAut(DID_i, PK_c, Token'_i, proof'_i) \rightarrow (0, 1))}$**.**
   1) After receiving the NIZK $proof'_i = \left(c'_i, s'_{1,i}, s'_{2,i}\right)$, if C wants to use the DU_i's ID to query personal information, they must send the relevant authorization parameters $(DID'_i, PK_c, Token'_i, proof'_i)$ to S for authorization verification, so that S can check whether the DU'_i queried by C and the DU_i authorized to C have the same sk_i and ID_i. the S performs the following verification operations:
      - Compute $Q'_{1,i} = DID'_i \cdot c'_i + G \cdot s'_{1,i} + H \cdot s'_{2,i}$;
      - Compute $Q'_{2,i} = Token'_i \cdot c'_i + PK_c \cdot s'_{1,i}$;
      - Compute ${c_i}^* = H_1 \left(G \,\|H\|\, DID'_i \,\|N\|\, Token'_i || PK_c || Q'_{1,i} || Q'_{2,i}\right)$;
      - The server verifies whether the equation $c'_i = {c_i}^*$ is true. If true, the verification passes, otherwise the verification fails.
   2) If the verification is passed, S performs subsequent operations; if the verification fails, the retrieval request operations are refused. The number of times this process is executed is determined by the number of keywords (n_1) queried by the C.

GenKS Algorithm:$\boldsymbol{(GenKS(Token, I, SK_c, sk_s) \rightarrow e)}$**.** This algorithm is inspired by the construction of mqRPMT and the C uses the $Token$ as the keywords to search the associated payloads. After running, the C outputs a choice bit vector $\mathrm{e} = (\mathrm{e}_1, \ldots, \mathrm{e}_{\mathrm{n}_1})$ such that $e_i = 1$ if and only if $ID_i = ID'_i$ but without knowing ID'_i, while the S learns nothing.

1) C inputs a set $Token = \{Token_1, \ldots, Token_{n_1}\}$ and the S inputs a set $I = \{(UID_1, D_1) \ldots, (UID_{n_2}, D_{n_2})\}$, where $D_i \in \{0, 1\}^l$.

2) S computes $F_{sk_s}(UID_i) = sk_s \cdot UID_i, i \in [n_2]$ with a sk_s and $M_i = H_2(F_{sk_s}(Token_i)) = H_2(sk_s \cdot Token_i), i \in [n_1]$.
3) S picks a random permutation π over $[n_2]$ and sends $M_{\Pi(i)} = H_2(F_{sk_s}(Token_{\Pi(i)})) = H_2(sk_s \cdot Token_{\Pi(i)}), i \in [n_1]$ and $F_{sk_s}(UID_i) = sk_s \cdot UID_i, i \in [n_2]$ to C. Another option is to add $M_i = H_2(F_{sk_s}(Token_i)) = H_2(sk_s \cdot Token_i), i \in [n_1]$ to a Bloom filter Ω instead of shuffling explicitly. Then S sends the resulting filter Ω' to the C.
4) For $i \in [n_2]$, C computes $M'_i = H_2(F_{SK_c,sk_s}(UID_i)) = H_2(SK_c \cdot sk_s \cdot UID_i)$ and if $\exists i \in [n_1], s.t. M_i = M'_i$ or $M'_i \in \Omega'$, they set $e_i = 1$, else set $e_i = 0$.

PayTrans Algorithm: ($\boldsymbol{PayTrans(e, I) \rightarrow D'_i}$). The two parties invoke the F_{OTE} algorithm to negotiate the symmetric key, in which e is used as the selection bits. Then the server uses symmetric encryption to transmit the payloads (corresponding to the e) to the client.

1) S initializes a random vector $s \in \{0,1\}^k$ and C a random $n_1 \cdot k$ bit matrix T.
2) C as a sender with inputs $(t_i, e \oplus t_i), i \in [k]$ and S acts as a receiver with input s. The parties invoke k times OT primitive.
3) let Q denote the $n_2 \cdot k$ matrix of values received by S. (Note that $q_j = (e_j \cdot s) \oplus t_i$.) For $j \in n_2$, S sends $(y_{j,0}, y_{j,1})$ where $y_{j,0} = D \oplus H_3(j, q_j)$, $y_{j,1} = D_{j,1} \oplus H_3(j, q_j \oplus s)$. The D is a random string of the same length as $D_{j,1}$.
4) For $1 \leq j \leq n_2 \cap e_j = 1$, C outputs $D'_i = y_{j,1} \oplus H_3(j, t_j)$.

Correctness.

- Given the PK_c, pp from running Setup algorithm and a tuple $(DID_i, token_i)$, the Completeness, Soundness and Zero-Knowledge of the NIZK $proof = (c_i, s_{1,i}, s_{2,i})$ is verified by the following:
  - Completeness: As long as the DU_i has the corresponding ID_i and sk_i, it can be verified by S through the following equations:

$$\begin{aligned} Q'_{1,i} &= DID_i \cdot c_i + G \cdot s_{1,i} + H \cdot s_{2,i} \\ &= (sk_i \cdot H + hid_i \cdot G) \cdot c_i + G \cdot \\ &\quad (r_{1,i} - c_i \cdot hid_i) + H \cdot (r_{2,i} - c_i \cdot sk_i) \\ &= r_{1,i} \cdot G + r_2 \cdot H = Q_{1,i} \end{aligned} \tag{9}$$

$$\begin{aligned} Q'_{2,i} &= Token_i \cdot c_i + PK_c \cdot s_{1,i} \\ &= hid_i \cdot PK_c \cdot c_i + PK_c \cdot (r_1 - c_i \cdot hid_i) \\ &= PK_c \cdot r_1 = Q_{2,i} \end{aligned} \tag{10}$$

$$c'_i = H_1\left(G \| H \| DID'_i \| N \| Token'_i || PK_c || Q'_{1,i} || Q'_{2,i}\right) = c_i \tag{11}$$

  - Soundness: If the DU does not have the corresponding ID and sk, it cannot pass the S's verification, that is, the probability of the prover deceiving the verifier is negligible. For detailed proof see Equations 9, 10 and Theorem 3.

- Zero-Knowledge: The DU only discloses to S whether he/she has the corresponding ID_i and sk_i statement (DID_i and $Token$ respectively), and uses the Pedersen Commitment and Shamir protocol to ensure that no additional information about the ID_i and sk_i will be leaked.

– Given the $SK_{c,i}$ and $sk_{s,i}$ from running GenKS Algorithm, and input a tuple set $(Token, I)$, the output e is correct except the event E that $F_{SK_c,sk_s}(UID_i) = F_{sk_s}(Token_{\Pi(i)})$ for some $ID' \neq ID$. Assuming $x \neq y$, by the collision resistance of H_2, we have $\Pr[H_1(ID') = H_1(ID)] = 2^{-k}$. The crucial observation is that

$$\begin{aligned} F_{sk_s,SK_c}(Token_i) &= H_2(sk_s \cdot (SK_c \cdot hid_i \cdot \alpha_i \cdot G)) \\ &= H_2(SK_c \cdot sk_s \cdot (hid_i \cdot \alpha_i \cdot G)) \\ &= F_{SK_c,sk_s}(UID_i) \end{aligned} \tag{12}$$

By pseudorandomness of F, we have $\Pr[E_i] = 2^{-k}$. Apply the union bound, we have $\Pr[E] = n_1 \cdot n_2 \cdot 2^{-k} = negl(\lambda)$

6.2 Security Proof

In this section, we provide detailed security proofs of Anonymity and Unforgeability of the proposed protocol. Moreover, we demonstrate that ASKPIR can achieve malicious security in the framework of Universal Composability (UC), which is described in Appendix A).

Theorem 2 (Data Subject Anonymity). *If Pedersen Commitment is hiding, the ASKPIR protocol satisfies anonymity of data subject.*

Proof. Suppose there exists a PPT $\mathcal{A}$ has a non-negligible advantage in $Expt_{ASPIR}^{Anon}(k)$, we can build a simulator B breaks hiding of Pedersen Commitment with the same advantage. B simulates $Expt_{ASPIR}^{Anon}(k)$ of definition 1 as follows:

- Setup: B executes the Setup algorithm, the system generates $pp \leftarrow SysGen(1^{\mathrm{k}})$, and sends pp to $\mathcal{A}$;
- Queries 1: Throughout the experiment, B responds $\mathcal{A}$'s queries as below:
  - $\mathcal{O}_{Reg}$: After receiving the query, B outputs honest DU_i's DID_i and responds it to $\mathcal{A}$;
  - $\mathcal{O}_{DID}$: After receiving the query, B access the DIDs on the Blockchain and return its to adversary $\mathcal{A}$;
- Challenge: $\mathcal{A}$ submits ID_0, ID_1 to B as the challenge values. B picks a random bit $0, 1 \overset{R}{\leftarrow} \beta$ and runs the Pedersen Commitment algorithm. B outputs DID_β and return it to adversary $\mathcal{A}$.
- Queries 2: $\mathcal{A}$ continues to adaptively query $\mathcal{O}_{Reg}$ and $\mathcal{O}_{DID}$.
- Guss: $\mathcal{A}$ outputs a guess β' for β and wins if $\beta = \beta'$.

Since DID_0 and DID_1 are two Pedersen Commitments, then suppose $\mathcal{A}$ can win the $Expt_{ASPIR}^{Anon}(k)$ with a non-negligible advantage, then the B solves the hiding of Pedersen Commitment with the same advantage. It contradicts the hiding of Pedersen Commitment and the advantage of an adversary against hiding is $Adv(k) = 0$, corresponding to Definition 3. Thereby, the advantage of $Adv_A^{Anon}(k) \leq negl(k)$, this proves the anonymity of data subject.

Theorem 3 (Unforgeability). *In ROM, the protocol is unforgeability of authorization if ECDLP is hard.*

Proof. Suppose there exists a PPT $\mathcal{A}$ has a non-negligible advantage in $Expt_{ASPIR}^{Unfor}(k)$, we can build a simulator B breaks the ECDLP with the same advantage. B controls the oracle and simulates an experiment $Expt_{ASPIR}^{Unfor}(k)$ of definition 1 as follows:

- Setup: Suppose that B gets a stochastic example $((Q, H, \gamma_1, \gamma_2) \in \mathbb{G}_p | \gamma_1 = x_1 \cdot G, \gamma_2 = x_2 \cdot H)$ of ECDLP problem and they want to compute x_1, x_2. The B inputs the challenge instance and runs $pp \leftarrow SysGen(1^{\mathrm{k}})$. The algorithm outputs pp, PK_c and sk_i and sends them to $\mathcal{A}$.
- Queries 1: B controls the ROM and completes the interaction between B and $\mathcal{A}$ as follows:
  - $\mathcal{O}_{Reg}$: After receiving the query, the B outputs honest DU's DID and responds it to $\mathcal{A}$;
  - $\mathcal{O}_{DID}$: After receiving the query, the B gets the $\{DID_1, \ldots, DID_n\}$,$\{PK_c, \ldots, PK_n\}$, $\{Token_1, \ldots, Token_n\}$ and $\{proof_1, \ldots proof_n\}$ on the block chain and return its to adversary $\mathcal{A}$;
  - H_s: B maintains list L of tuple $\{\delta_i, c_i\}$. When adversary $\mathcal{A}$ issues a query $\delta = (G, H, DID^*, N, UID^*, Token^*, pk^*, Q_1, Q_2)$, B checks whether it has been inquired before. If it has been inquired before, it finds the corresponding record and returns it to $\mathcal{A}$; Otherwise, B randomly selects C_i^* and adds $\{\delta_i, c_i^*\}$ to L.
- Forge: After receiving the query, B outputs a forged NIZK proof $(proof^* = (c^*, s_1^*, s_2^*))$. B first picks two randomly numbers $r_1, r_2 \in \mathbb{Z}_q$, and computers $Q_1 = r_1 \cdot G + r_2 \cdot H$ and $Q_2 = r_1 \cdot pk$ to $\mathcal{A}$. Then $\mathcal{A}$ inputs some parameters $(G, H, DID^* || N || Token^* || PK_c^* || Q_1 || Q_2)$ to ROM and return c_i^*.
- Queries 2: $\mathcal{A}$ continues to adaptively query $\mathcal{O}_{Reg}$ and $\mathcal{O}_{DID}$.

Solve ECDLP Problem: To generate the forged NIZK proof $(proof^* = (c_i^*, s_1^*, s_2^*))$ on the tuple $(G, H, DID^*, N, UID^*, Token^*, pk^*, Q_1, Q_2)$, $\mathcal{A}$ must query ROM and return c_i^*. Follow on, if B verifies that the NIZK $proof^*$ is invalid, interrupt interaction with the $\mathcal{A}$; Otherwise, from the Forking Lemma(Theorem 1), after a polynomial replay of the adversary $\mathcal{A}$, we obtain two valid NIZK proofs $proof' = (c', s_1', s_2')$ and $proof^* = (c^*, s_1^*, s_2^*)$ with $c' \neq c^*$. Then B has the following equalities: $x_1 = HID_i = (s_1' - s_{11}^*)/(c' - c^*)$ and $x_2 = sk_i = (s_2' - s^*{}_2)/(c' - c*)$, from which it obtain discrete logarithm of elliptic curves $G \cdot \gamma_1^{-1}$ and $G \cdot \gamma_2^{-1}$.

Therefore, B can solve the ECDLP with a ε probability, which contradicts the Assumption 1, so the advantage of $Adv_A^{Unfor}(k) \leq negl(k)$. Thereby, our protocol satisfies the authorization unforgeability.

7 Theoretical Comparison

Table 3 presents an evaluation of the theoretical computational and communication complexity of our protocol in compare with the protocols of Cristofaro et al. [6] and Layouni et al. [17,18]. We divide all protocols into three phase for detailed analysis, namely the initial stage, the authorization phase and the SPIR phase. The elliptic curve group elements are sized by ρ, with a value of 256 bytes. Additionally, for the exponentiation operations in cyclic group, there are $|p| = 1024$ bytes. l is the length of payloads ($l = 32$ in the concrete numbers). T_e is the exponentiation operation time in the cyclic group; T_m is the multiplication operation time in the cyclic group; T_{bp} is the execution time of a bilinear pairing operation; T_h is the execution time of a general hash function operation; $T_{m(ec)}$ is the execution time of a scale multiplication operation related to the ECC.

In terms of computational complexity, our protocol does not result in a significant increase in computation cost for authorization. Furthermore, when compared to other related works, our protocol exhibits notable advantages in terms of computational overhead. (1) In the Setup phase, the protocol of Cristofaro et al. [6] and Layouni et al. [17] must generate RSA-based keys, which requires a lot of computing resources. Besides, the computational overhead of Layouni et al. [17] is also related to the l, which is not suitable for transmitting large messages. The computing cost of Layouni et al. [18] is mainly related to the number of data subjects n_m that need to be authorized, and the computing cost is up to $O(n^2)$. In our protocol, the system generates keys simply and DU can quickly calculate and generate DID, which is mainly completed by $T_m(ec)$. (2) In the Authorization phase, the computational cost of Layouni et al. [17] is related to the DU's anonymous credential, which involves a lot of T_e operations.The computational overhead of Layouni et al. [17] protocol and Cristofaro et al. [6] protocol is similar, mainly consisting of T_e operations. Our protocol is the lowest because we mainly spend $T_m(ec)$ computational overhead in building the authorization algorithm. Furthermore, our protocol generates $Tokens$ during authorization and performs precomputation for client input, reducing the computing overhead during the next phase. (3) In the SPIR phase, the computational cost is mainly related to n_2. The protocol of Layouni et al. [18] performs a large number of bilinear operations T_{bp} and uses asymmetric encryption to transmit data, with a computational cost of up to $O(n^2)$. The computational cost of Layouni et al. [17], Cristofaro et al. [6] and our protocol is mainly related to T_e, T_m and $T_m(ec)$ respectively, among which $T_m(ec)$ requires the least computing resources. Moreover, our protocol uses OTE technology, which avoids public key calculation and can form symmetric encryption to transmit data. Based on the above analysis, our protocol performs best in terms of computational consumption.

In terms of communication complexity, our protocol performs better and requires less communication bandwidth overhead. (1) In the Setup phase, our protocol does not require registration with a third party and does not consume bandwidth by interacting with any party. the protocol of Layouni et al. [17] needs to interact with CA to generate attribute-based anonymous credential, and bandwidth consumption is also related to l. The computational overhead of Layouni et al. [18] protocol and Cristofaro et al. [6] protocol is similar, mainly consisting of $n_1|p|$. (2) In the Authorization phase, the Layouni et al. [17] protocol involves multiple rounds of interactions between DU and the client in the process of issuing anonymous credentials, and the communication data is generated by T_e and T_h operations. The protocol of Layouni et al. [18] and Cristofaro et al. [6] have fewer rounds of interaction, and the main communication data is generated by T_e. As for our protocol, the bandwidth consumption is mainly related to the length of the hash value. (3) In the SPIR phase, the two protocols of Layouni et al. [17,18] rely heavily on asymmetric encryption, the communication cost of transmitting ciphertext is high, and they are not suitable for batch retrieval of large data payloads. In the SPIR phase, the two protocols of Layouni et al. [17,18] rely heavily on public key encryption, and the communication cost of transmission is mainly related to the value of $|p|$. The protocol of Cristofaro et al. [6] performs best, with only two rounds of interaction and communication complexity independent of the client's query set n_1. A large number of matrix transfers mainly causes our bandwidth resource consumption. Still, the Setup and Authorization phases save a lot of communication overhead, especially since ECC-based protocol has low bandwidth requirements and can be easily piped. Our protocol can run on millions of elements even on a standard PC.

Table 3: Comparison of Theoretical Computational and Communication Complexity of Related Works.

Protocol	Computational		
	Setup	Authorization	SPIR
[6]	$n_1(T_e + T_h)$	$n_1(3T_e + T_m + T_h)$	$n_1(T_e + 2T_h) + n_2(T_m + T_h)$
[17]	$n_1 l T_e$	$n_1((l+1)T_e + T_m + 2T_h)$	$3n_1 T_e + 3n_2 T_e$
[18]	$n_m n_1 T_e$	$n_1(2T_e + T_h)$	$n_1 n_2 T_p + n_2 T_m$
ASKPIR	$T_{m(ec)} + n_1(T_h + 2T_{m(ec)})$	$n_1(8T_{m(ec)} + 2T_h)$	$n_1 T_h + n_2(T_{m(ec)} + 3T_h)$

Protocol	Communication		
	Setup	Authorization	SPIR
[6]	$3n_1\lvert p\rvert$	$n_1\lvert p\rvert$	$n_2(l + \lambda + \log_2{}^{n_1 n_2} + \lvert p\rvert)$
[17]	$n_1 l\lvert p\rvert$	$n_1(\lvert p\rvert(6+2l) + \lambda + \log_2{}^{n_1 n_2})$	$4n_1\lvert p\rvert + 2n_2\lvert p\rvert$
[18]	$n_1\lvert p\rvert$	$3n_1\lvert p\rvert$	$3n_1\lvert p\rvert + n_2(\lambda + \log_2{}^{n_1 n_2} + l\lvert p\rvert)$
ASKPIR	$\rho(1+2n_1)$	$n_1(\rho + 2\lvert N\rvert + \lambda + \log_2{}^{n_1 n_2})$	$n_1(1.2k) + n_2(\lambda + \log_2{}^{n_1 n_2} + \rho + 2l)$

8 Performance Evaluation

8.1 Security Attribute and Function Evaluation

This section examines the security attribute and function of our protocol and compares it to related works. In Table 4, we list several features of the protocols, such as Anonymity (T_1), Unforgeability (T_2), Decentralized registration (T_3), Batch Retrieval (T_4), Collusion Attack (T_5) , Malicious Model (T_7) and Symmetric encryption (T_7).

Our protocol is based on DID, which provides an effective method for data subjects to authorize data and maintain identity privacy in a distributed environment, and guarantees the anonymity of the data subject's identity and the

Table 4: Comparison of Security Attribute and Function of Related Works.

Protocol	T_1	T_2	T_3	T_4	T_5	T_6	T_7
[6]	✓	×	×	✓	✓	×	✓
[17]	✓	✓	×	×	×	×	×
[18]	×	✓	×	×	×	×	×
ASKPIR	✓	✓	✓	✓	✓	✓	✓

unforgeability of the authorization information. Additionally, the protocol builds PSI-Payload to enable batch keyword searches and Transmit payload with symmetric encryption. The Layouni et al. [17] protocol offers anonymous credentials to ensure both anonymity and unforgeability, but it only considers single-user authorization. Besides, the protocol introduces a third-party CA that may collude with the server and cannot resist collusion attacks. Fortunately, our protocol does not require registration with a third party and is resistant to collusion attacks. The Lajuni et al. [18] protocol enables multiple data subjects to authorize client to query data. However, the asymmetric encryption used in data transmission is very complex and cannot be applied to uniformly manage the identities of multiple data subjects and ensure their anonymity in a distributed environment. This protocol only hides the ID of data subjects in exponential operations and does not uniformly manage IDs. Moreover, the server authorization is only to verify whether the client's access data complies with its defined policy, and does not consider the unforgeability of the authorization. Overall, our protocol compensates for all the shortcomings of existing protocols.

8.2 Experimental Results

We develop ASKPIR on the basis of Curve25519 [3], where $a = 115792089210356248762697446949407573530086143415290314195533631308867097853948$, $b = 41058363725152142129326129780047268409114441015993725554835256314039467401291$, $p = 2^{255}-19$, and $N = 57896044605178124381348723474703786765043071707645157097766815654433548926975$ and Our PSI-Payload is developed based on the $mqrpmt_{psi}$ code in Kunlun library [30]. All our experiments were implemented on a benchmark machine with Intel Core $i5$ 2.4 GHz, 32 GB RAM, 8 physical cores. 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 100 MB network bandwidth was used. The server held the input sets in $n_2 \in \left\{2^{12}, 2^{16}, 2^{20}\right\}$. The comparison measures how much communication a protocol would require over an idealized network, regardless of other environmental and encoding factors.

In Table 5, we give detailed computational and communication performance results for 4 threads execution. For the server size $n_2 = 2^{20}$ and client set size $n_1 = 2^8$, we obtain an overall running time of 31.56 s and only 97.47 MB of

Table 5: Communication cost and running time of our protocol.

n_2	n_1	Communication (MB)				LAN(s)			WAN(s)		
		Authn	Client	Server	Total	Authn	Authz	Total	Authn	Authz	Total
2^{12}	2^8	0.4	0.32	0.099	0.8224	1.1	0.25	1.53	6.4	0.85	9.14
	2^{10}	1.8	0.144	0.32	2.2	15.72	0.28	16.24	67.1	0.97	69.27
	2^{12}	8.6	0.33	0.32	9.25	255.36	1.69	264.14	548.29	2.56	565.15
2^{16}	2^8	0.45	1.032	5.07	6.15	10.09	0.63	13.12	8.32	0.95	22.37
	2^{10}	1.89	1.082	5.07	7.962	15.2	0.34	23.94	38.2	1.24	74.44
	2^{12}	7.6	1.27	5.081	13.96	207.44	0.56	210.5	457.2	1.56	462.36
2^{20}	2^8	0.47	16	81	97.47	1.4	0.15	31.56	2.21	0.89	53.1
	2^{10}	1.89	16.082	81	98.972	18.2	1.69	59.89	34.23	2.99	107.22
	2^{12}	7.6	16	81	104.6	267.36	1.89	307.25	350.54	2.26	407.8

communications within a LAN, in which the running time of 1.4 s and the communication cost of 0.47 MB for the authorization (Authn). In addition, the total time it takes for the server to verify whether the data subjects' authorization is valid (Authz) only takes 0.15 s. When considering the medium server set size $n_2 = 2^{16}$ and client set size $n_1 = 2^8$, our protocol requires only 6.15 MB of communication and 22.37 s of total time of communications within a WAN, in which the running time of 8.32 s and the communication cost of 0.45 MB for the authorization (Authn). The total time to verify the authorization of the server only takes 0.15 s. When considering the small server set size $n_2 = 2^{12}$ and client set size $n_1 = 2^{10}$, our protocol requires only 2.2 MB of communication and 16.24 s of total time of communications within a LAN, in which the running time of 15.72 s and the communication cost of 1.8 MB for the authorization (Authn). The total time to verify the authorization of the server only takes 0.28 s. It can be seen that in the first two cases, the authorization operation does not affect the communication and computing costs, and the performance is only related to the size of the client. When the input set of the server is small, since we use ECDH-based PSI-Payload in the SPIR phase, the computational cost and communication overhead of the protocol are not high, which is quite different from the authorization overhead determined by the size of the client set. However, in real scenarios, DIDs are pre-generated by the data subject, and the time it takes for the data subject to generate authorization is shorter than our experimental results. The implementation also takes advantage of our authorization phase to perform a precomputation to reduce the online running time of SPIR phase. We remark that our pre-processing can be done entirely offline without involving the client. Specifically, the $Token$ issued by the data subjects to the client during the authorization phase can not only participation in NIZK and be verified by the server, but can also be used as an input set by the client, saving a lot of computing overhead of client. It is worth noting that the computational workload for the client is minimal, involving only the sampling of random values and a scale multiplication operation that is associated with the ECC. Moreover, all the operations are performed in a streaming manner, allowing for the greatest amount of work to be carried out concurrently by the parties. Therefore, our reported performance is also indicative of a scenario in which the client is a less powerful device connected via a mobile network.

9 Conclusion

The focus of this paper is to solve the authorization problem of SPIR protocol. More precisely, the protocol allows the client to use the identity of the data subject to query personal information before the server needs to verify that the identity information has been authorized by the data subject. At the same time, the identity of the data subject cannot be learned by the server. The structure we proposed mainly relies on DID, which not only realizes the identity management and privacy of multiple data subjects when sharing data in a distributed environment, but also solves the data authorization problem of data subjects in combination with NIZK. On this basis, we introduced the functions of batch retrieval based on PSI-Payload. Eventually, we implement this construct and propose a new SPIR primitive, called ASKPIR. The proposed construction is proved secure under the malcious environment and contain anonymity and unforgeability. Even with the expanded authorization, the presented protocol does not introduce greater complexity than that of the underlying SPIR.

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. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 673–701. Springer (2015)
3. Bernstein, D.J.: Curve25519: new diffie-hellman speed records. In: Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9. pp. 207–228. Springer (2006)
4. 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
5. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings (2004)
6. 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)

7. De Cristofaro, E., Lu, Y., Tsudik, G.: Efficient techniques for privacy-preserving sharing of sensitive information. In: International Conference on Trust and Trustworthy Computing. pp. 239–253. Springer (2011)
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. Garzon, S.R., Yildiz, H., Küpper, A.: Decentralized identifiers and self-sovereign identity in 6g. IEEE Network **36**(4), 142–148 (2022). https://doi.org/10.1109/MNET.009.2100736
10. 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)
11. Goyal, P.: Private Information Retrieval with Access Control. Ph.D. thesis, Massachusetts Institute of Technology (2023)
12. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Annual International Cryptology Conference. pp. 145–161. Springer (2003)
13. 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)
14. 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)
15. Kissner, L., Oprea, A., Reiter, M.K., Song, D., Yang, K.: Private keyword-based push and pull with applications to anonymous communication. In: Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004. Proceedings 2. pp. 16–30. Springer (2004)
16. Kolesnikov, V., Rosulek, M., Trieu, N., Wang, X.: Scalable private set union from symmetric-key techniques. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 636–666. Springer (2019)
17. Layouni, M.: Accredited symmetrically private information retrieval. In: Advances in Information and Computer Security: Second International Workshop on Security, IWSEC 2007, Nara, Japan, October 29-31, 2007. Proceedings 2. pp. 262–277. Springer (2007)
18. Layouni, M., Yoshida, M., Okamura, S.: Efficient multi-authorizer accredited symmetrically private information retrieval. In: Information and Communications Security: 10th International Conference, ICICS 2008 Birmingham, UK, October 20-22, 2008 Proceedings 10. pp. 387–402. Springer (2008)
19. 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)
20. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Crypto. vol. 99, pp. 573–590. Springer (1999)
21. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)

22. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE symposium on security and privacy. S&P 2000. pp. 44–55. IEEE (2000)
23. Stinson, D.R., Strobl, R.: Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In: Australasian Conference on Information Security and Privacy. pp. 417–434. Springer (2001)
24. 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
25. 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)
26. 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)
27. Wang, Z., Ulukus, S.: Digital blind box: Random symmetric private information retrieval. In: 2022 IEEE Information Theory Workshop (ITW). pp. 95–100. IEEE (2022)
28. Yoshida, R., Cui, Y., Shigetomi, R., Imai, H.: The practicality of the keyword search using pir. In: 2008 International Symposium on Information Theory and Its Applications. pp. 1–6. IEEE (2008)
29. 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)
30. Zhang, C., Chen, Y., Liu, W., Zhang, M., Lin, D.: Linear private set union from {Multi-Query} reverse private membership test. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 337–354 (2023)

A Security Proof in Malicious Mode

Theorem 4. *The ASKPIR protocol realizes the functionality of $F_{PSI}^{Payload}$ against a malicious adversary in the random oracle, F_{mqRPMT} and F_{OTE}-hybrid model.*

Proof. First observe that the protocol is correct. We prove the following two Lemmas:

Lemma 1. *The ASKPIR protocol realizes the functionality of $F_{PSI}^{Payload}$ against a malicious client in the ROM, F_{mqRPMT} and F_{OTE}-hybrid model.*

Proof. Consider a malicious client $\mathcal{A}$. The simulator B plays the role of F_{mqRPMT} and F_{OTE}, and interacts with the client as follows:

1) Client sends $X = (Token_1, \ldots, Token_{n_1})$ to the F_{mqRPMT}, the simulator B observes X and sends e back as the F_{mqRPMT} would.
2) B observes all of the client's queries to random oracle H_1.
3) B computes $ID^* = \{Token^* = SK_c \cdot \alpha_i \cdot H_1(ID_i^*) \cdot G | ID_i^*$ was queried to $H_1\}$, and sends this to $F_{PSI}^{Payload}$ as the client's effective input.

4) Upon receiving from the ideal functionality the intersection payload $\mathrm{Z}^* = \{\mathrm{D_i} \,|\mathrm{ID_i}= \mathrm{ID}_i^*, 1 \leq \mathrm{i} \leq \mathrm{n_1}\}$, the simulator simulates the server's input set Y^* as $\{(UID_i^* = H_1(ID_i^*) \cdot G, D^*)|D^* \in \mathrm{D}'\}$ along with $|n_1| - |e|$ additional random values.

To demonstrate the indistinguishability of this simulation, consider the following hybrid scenarios:

Hybrid 1: The same as the real protocol except the the simulator keeps a list L of all queries directed to the random oracle H_1 by the adversary $\mathcal{A}$. When the adversary selects its F_{mqRPMT} input $X = (Token_1, \ldots, Token_{n_1})$, the simulator checks all $ID' \in L$ and defines the set $ID^* = \{Token^* = SK_c \cdot H_1(ID_i^*) \cdot G | ID_i^* \text{ was queried to } H_1\}$. This hybrid remains indistinguishable from the actual protocol interaction, as the only difference lies in internal record-keeping data that is not utilized.

Hybrid 2: Now we change the previous Hybrid: after defining ID^*, the simulator aborts if the honest server holds an $\mathrm{UID_i} = \mathrm{H_1}(\mathrm{ID_i'}) \cdot \alpha_\mathrm{i} \cdot \mathrm{G}$ where $ID_i' \notin ID_i^*$. It is adequate to demonstrate that the probability of this artificial termination is negligible.

- Case $ID_i' \in L$: then $H_1(ID_i)$ was known at the time UID was defined. Therefore it is by construction that $ID_i \in \mathrm{ID}^* \Leftrightarrow UID_i = H_1(ID_i) \cdot \alpha_i \cdot G$. In other words, an abortion does not occur in this scenario.
- Case $ID_i \notin L$: then $H_1(ID_i)$ is independent of UID_i, and thus $UID^* = H_1(ID_i^*) \cdot \alpha_i \cdot G$ with probability $2^{-l_1 \cdot N}$.

If $l_1 = k + \log_2{}^{n_2 \cdot n_1}$ then by a union bound over at most n_2 possible server's values, the probability of an abortion is indeed limited to 2^{-k}.

Hybrid 3: The UID in previous experiment is modified and the simulator samples UID. The simulator in this hybrid aborts if any $F_{sk_c}(UID_i)$ has been made by the adversary $\mathcal{A}$. Since UID_i is randomly sampled, each $sk_s \cdot UID_i$ follows a uniform distribution, thereby ensuring that the probability of abortion is at most $\mathrm{O}(2^{-k})$.

Hybrid 4: Similar to **Hybrid 3**, we have the ability to modify the computation that defines the server's $F_{sk_s}(UID_i)$ message. Observe that

$$\begin{aligned} &F_{sk_c}(UID_i) = \{H_2(sk_s \cdot UID_i)|i \in [n_1]\} \\ &= \{H_2(sk_c \cdot \alpha_i \cdot H_1(ID_i) \cdot G)|i \in [n_1]\} \end{aligned} \tag{13}$$

Due to the artificial termination introduced in the prior hybrid, this happens for $ID_i' \in ID^*$ if and only if $ID' \cap ID^*$. Hence, we can rewrite the server's $F_{sk_s}(UID_i)$ message as:

$$\begin{aligned} &F_{sk_s}(UID_i) = \{H_2(sk_s \cdot UID_i)|UID_i \in X\} \\ &= \{H_2(sk_s \cdot \alpha_i \cdot H_1(ID_i') \cdot G)|ID_i' \in ID' \cap ID^*\} \\ &\cup \{H_2(sk_s \cdot \alpha_i \cdot H_1(ID_i') \cdot G)|ID_i' \in ID' \backslash ID^*\} \end{aligned} \tag{14}$$

where the $\{H_2(sk_s \cdot \alpha_i \cdot H_1(ID_i') \cdot G)|ID_i' \in ID' \backslash ID^*\}$ values are guaranteed to be random. This hybrid is indistinguishable from the previous one, as we have merely reformulated the same computation equivalently.

Hybrid 5: Similar to **Hybrid 4**, except the the simulator no longer artificially terminates the process as introduced in **Hybrid 2**. The indistinguishability of the hybrids is upheld for the same reasons as previously explained. In this case, the simulator does not utilize the items from $\{ID' \backslash ID^*\}$ at all. We conclude the proof by observing that this hybrid exactly describes the final ideal-world simulation: the simulator extracts ID^*, sends it to the ideal PSI functionality, receives Z^* , and uses it to simulate the server's message $F_{sk_s}(UID_i)$. We conclude the proof by noting that this hybrid precisely character the final ideal-world simulation. The simulator extracts ID^*, transmits it ideal PSI-Payload functionality, receives Z^*, and employs it to simulate the server's message $F_{sk_s}(UID_i)$.

Lemma 2. *The ASKPIR protocol realizes the functionality of $F_{PSI}^{Payload}$ against a malicious server in the ROM, F_{mqRPMT} and F_{OTE}-hybrid model.*

Proof. Consider a Malicious Server. The simulator plays the role of F_{mqRPMT} and interacts with the server as follows:

1) It observes the server's input $I = \{UID_1, \ldots, UID_n\}$ and output $\{M_i, i \in [n]\}$ from F_{mqRPMT}, and also observes all of the server's queries to random oracle H_2. Let I^* be the set of all such UID_i.
2) When $\mathcal{A}$ sends $M_i = H_2(F_{SK_c sk_s}(UID_i))$, the simulator computes $I^* = \{UID | UID \in I^* \wedge \neg \exists UID' \in I^* s.t. UID \neq UID' \wedge F_{sk_s SK_c}(UID) = F_{sk_s SK_c}(UID')\}$ and extracts $I' = \{UID | UID \in I^* \wedge H_2(F_{sk_s SK_c}(UID)) \in M_i\}$ and sends I' to $F_{PSI}^{Payload}$.

To establish the indistinguishability of this simulation, consider the following hybrid scenarios:

Hybrid 1: In this hybrid, which is identical to the actual protocol, the simulation takes on the role of F_{mqRPMT}.

Hybrid 2: Same as the real protocol interaction, except that the simulator observes the server's input $I = \{UID_1, \ldots, UID_n\}$ and output $\{M_i, i \in [n]\}$ for F_{mqRPMT}, and additionally observes all queries made to random oracle H_2. The simulator defines a set L consisting of all values $F_{sk_s, SK_c}(UID)$ for which the adversary queried H_2 using the "correct" value $F_{sk_s, SK_c}(UID)$. Upon receiving protocol message M, the simulator further defines the set $I^* = \{UID | UID \in I^* \wedge \neg \exists UID' \in I^* s.t. UID \neq UID' \wedge F_{sk_s SK_c}(UID) = F_{sk_s SK_c}(UID')\}$. This hybrid scenario is indistinguishable from real protocol interaction, as it solely involves recording bookkeeping information that remains unused.

Hybrid 3: Same as **Hybrid 1**, except the simulator aborts if the honest client holds $\{Token_i = UID_i {\cdot} SK_c | UID_i \in I \backslash I^*\}$ where $M_i' = H_2(F_{sk_s}(Token_i)) \in M$. There are two cases for why such a $Token_i$ may not be in I^*:

- Case $F_{sk_s}(Token_i) \in L$: then the value $H_2(F_{sk_s}(Token_i))$ was defined at the time I^* was computed, and $Token_i$ was excluded because the correct value was not in M'. The simulator will never abort in this case.
- Case $F_{sk_s}(Token_i) \notin L$: $\mathcal{A}$ did not query H_2 at $H_2(F_{sk_s}(Token_i))$ prior to sending M', thus the output of H_2 is uniformly random and independent

of M'. The probability that this H_2 output appears in M is thus $|M|/2^{l_2}$ where l_2 is the output length of H_2.

Overall, the probability of such an artificial abort is bounded by $n|M|/2^{l_2} \leq n^2/2^{l_2}$. Hence the two hybrids are indistinguishable.

Hybrid 4: The previous hybrid was modified in this study to compute the output of the honest client. In the **Hybrid 2**, the honest client computes the output according to the protocol specification as follows: $\{H_2(F_{sk_s}(Token_i)) \in M\}$. In this hybrid approach, we compute the output of the honest client as $e^* = \{e_i | ID^* = ID\}$. The two expressions are indeed equivalent, based on the definition of I^* and the introduction of artificial abort in the previous expression. Furthermore, the equivalence between $H_2(F_{sk_s}(Token_i))$ and $F_{sk_s SK_c}(UID)$ discussed in the previous proof also holds.

Hybrid 5: Same as **Hybrid 4**, except we remove the artificial abort condition that was introduced in **Hybrid 3**. Similar to **Hybrid 4**, we eliminate the artificial abort condition introduced in **Hybrid 3**. The indistinguishability of the hybrids remains unchanged for the same reasons as before. It is important to note that in this hybrid, the simulator solely utilizes the honest client's input I for computing their final output. By considering this hybrid, we can precisely describe the simulation of an ideal world: The simulator observes both I and M, which represents server's oracle queries, to derive a set I^*. Subsequently, it sends I^* to the ideal functionality resulting in delivery of e^* to the client.