Towards more secure constructions of private set operation schemes

Mojtaba Rafiee[1,2*]

[1*]Department of Applied Mathematics and Computer Science, Faculty of Mathematics and Statistics, University of Isfahan, Isfahan, Iran.

[2]School of Mathematics, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

Corresponding author(s). E-mail(s): m.rafiee@mcs.ui.ac.ir;

Abstract

A private set operation (**PSO**) scheme [Rafiee, Comput. J. 2020] is a cryptographic primitive that enables a user to securely outsource their dataset to cloud storage, and then when needed, securely issue common set operation queries to the server and receive the results. In [Rafiee, Comput. J. 2020], the only security notion of the **PSO** schemes, named **naSIM**, is proposed. This security notion models a weak attacker who is far from the threats of practical environments, and providing stronger security notions has been raised as an open problem. In this paper, we propose a new security notion for the **PSO** schemes, called **aIND**, and show that this concept is stronger than **naSIM**. Furthermore, we propose a new **PSO** construction that satisfies the security notion **aIND**. We also show that our construction does not increase the computational and storage overheads compared to other existing constructions, despite covering a much higher level of security.

Keywords: Secure outsourcing, Cloud storage, Cryptographic primitive, Private set operation, Adaptive security

1 Introduction

In recent years, the popularity and use of cloud services, specifically computing and storage services, has increased. In the settings of such services, a user with limited computing and storage resources outsources his data and computations to a cloud service that, from his point of view, has unlimited computing and storage resources. Numerous cloud services, such as database management services, document management and storage services, online markets, and security services, were developed according to the needs of society and industry. In such services, since the data, as an important value of any person or organization, is managed and stored away from its owner and on the storage space of an external server, data confidentiality protection is of great importance. Therefore, the use of cryptographic primitives is essential to provide such a security feature.

In this regard and in recent years, several cryptographic primitives have been proposed to support various functionalities in cloud services, such as:

- **Searchable Symmetric Encryption** (SSE) **schemes** [1, 2]**:** These schemes enables a user to encrypt a collection of documents in a way that he can later efficiently and securely search over the encrypted data.
- **Tuple Set** (Tset) **schemes** [3]**:** These schemes enables a user to securely store a hashed form of a datatable and to generate a token for each row. The token can then be used to retrieve the corresponding row publicly from the hashed datatable.
- **Adjustable Join** (Adjoin) **schemes** [4]**:** These schemes enables a user to securely outsource his database and later issue join queries for any pair of columns of interest in the database.
- **Private Set Operation** (PSO) **schemes** [5]**:** These schemes enables a user to securely outsource their dataset to cloud storage, and then when needed, securely issue common set operation queries to the server and receive the results.

The mentioned cryptographic primitives have been widely used in the design of protocols for secure cloud services. However, among these schemes, the PSO schemes have always received special attention due to their support for set operations, which are known as the most basic mathematical operations and a large number of functions can be defined based on them. For this reason, in this paper, we study PSO schemes and try to develop its security concepts and present a new PSO construction that satisfies it.

Remark 1 It should be noted that the PSO schemes considered in this paper are different from the well-known private set operation schemes defined in the Multi-Party Computation (MPC) framework. The basic scenario model for PSO schemes in MPC includes several parties who each locally holds a private set. These parties interact with each other and compute a set operation. In such a setting, the goal is to compute the result in such a way that none of parties is able to gain any additional information besides what can be inferred from their own local set and the computed

Table 1: A feature-based comparison between PSO constructions.

Construction	Cryptographic tools		Security properties	
	Tset	Adjoin	Hardness assumption	Security level
PSO I	Tset [3]	Adjoin from [8]	matix-DDH [9]	Leak-naSIM
PSO II		Adjoin from [5]	XDHV [5]	Leak-naSIM
Our PSO		Adjoin from [4]	DDH1 [10]	Leak-aIND

result [6, 7]. Such a level of security makes the protocols ineffective for real-world applications. This is while the scenario model and the security notion of the PSO schemes in [5] are aimed to reduce computing, storage and communication overheads and to be close to real-world uses.

For the first time, the syntax and security notions of the PSO schemes were proposed by Rafiee et al. in [5]. In particular, they presented a simulation-based security notion for PSO schemes, called naSIM, and for the first time introduced a reasonable leakage profile for such schemes in the form of a leakage function. It should be noted that the naSIM security notion is the only security definition available for PSO schemes at the time of writing this paper, and also all the available proposed PSO constructions with naSIM security are limited to paper [5]. All of these constructions are based on a generic PSO construction and are distinguished from each other based on the different cryptographic constructions used for its building blocks.

In this paper, we introduce a stronger security notion than the naSIM security notion, called aIND, and propose a new PSO construction based on the generic PSO construction that covers it. Table 1 shows a feature-based comparison of existing PSO constructions based on the cryptographic primitives used (Tset and Adjoin) and security properties (computational hardness assumption and achieved security level).

Challenges and contributions. As stated earlier, the only security notion available for PSO schemes is naSIM. In fact, this security notion models a non-adaptive attacker. Such an attacker is unrealistic for practical environments. The reason is that we restrict the attacker to only issuing all of his queries (in the outsource and query phases) at once and in one go. In such a situation, the attacker has limited power to challenge PSO schemes, and thus provides a low level of security (see Subsection 4.2). In this paper, we explore and introduce a stronger security concept in which the attacker's adaptive queries are modeled, called aIND. Such an attacker challenges the PSO schemes gradually and purposefully by sending queries and receiving responses. The definition of such a security notion is raised as an open problem in paper [5].

As we know, proposing a new security notion alone does not prove its validity, and this security concept is considered valid when:

- Reasonably defined for the practical environments. This item means that the desired security features are properly included, as well as constraints

that prevent the attacker from obviously succeeding. To cover this feature, we have used standard and basic concepts available in the literature.
- In addition to the above, a construction must be provided that satisfies the security concept proposed. This item is known as one of the most challenging parts of presenting a new security concept. It should be noted that when we want to keep its efficiency at the level of other existing constructions, its level of challenge increases significantly.

In this regard, we propose a new PSO construction that satisfies our new security concept. In the theoretical cryptography literature, proposing a new construction with higher security level for the first time is highly valuable, regardless of its efficiency. This is while the trade-off principle between security and efficiency also confirms that increasing the security level of a scheme will usually lead to a decrease in its efficiency, and vice versa. In such a situation, our goal is to cover the stronger security level aIND for our PSO construction, while maintaining the efficiency at the level of other existing PSO constructions.

Some of the most important steps we took to achieve the above goals are as follows:

- The definition of a new security concept that models adaptive adversaries, and that is developed in accordance with previous standard definitions in the literature (see Subsection 5.1).
- State the implications and separations between our new security notion and previous security notions for the PSO schemes (see Subsection 5.2).
- Using appropriate cryptographic primitives as the building blocks for our new construction (see Section 6).
- Using cryptographic techniques and lemmas to provide provable security for our PSO construction (see Section 6).
- Performance analysis of our PSO construction in comparison with other existing PSO construction, asymptotically and concretely (see Section 7).

Paper organization. The remaining of this paper is organized as follows. The scenario model, threat model and problem statement are described in Section 2. In Section 3, we provide the notations, cryptographic concepts and basic cryptographic primitives that are required throughout this paper. Section 4 reviews the syntax and security notions of the PSO schemes, as well as the generic construction provided for them. A new security notion for the PSO schemes and a construction that satisfies it are presented in Sections 5 and 6, respectively. The performance analysis of our construction is provided in Section 7. Finally, Section 8 concludes the paper and points out future directions.

2 System model

In this section, we study our desired system model. In this regard, we first describe the system entities and their interactions with each other. In the

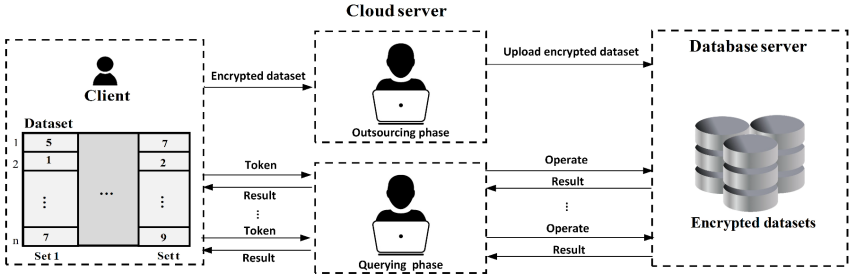


Fig. 1: Overview of the system [5].

following, we determine the trust level for each entity in the system. Finally, we explain the problem statement in the described scenario model.

2.1 Scenario model

Our desired scenario model includes the following two entities:

- **User:** an entity that, due to computational and storage limitations, has outsourced its dataset to an external server.
- **Cloud service provider:** an entity that has a lot of computing and storage resources and, in our model scenario, is responsible for managing, maintaining, and executing set queries on outsourced datasets.

The interaction between the mentioned entities is formed in the following two phases:

- **Outsource phase:** this phase is performed only once, at the beginning of the user-server interaction, during which the encrypted dataset is placed on the cloud storage server.
- **Query phase:** this phase is executed according to the user's requested queries, and the cloud server computes and returns the desired results to the user. It should be noted that unlike the outsourcing phase, which is only done once, the query phase is done multiple times.

Figure 1 summarizes the scenario model based on entities and how they interact with each other.

2.2 Threat model

In our scenario model, the user, since he owns the dataset and has access to its plain-dataset, is labeled as a fully trusted entity. However, the cloud server is labeled honest-but-curious because it is an external entity that keeps the dataset away from the user and on its cloud storage. An entity is called the honest-but-curious if it follows the scheme correctly, but plays the role of an eavesdropper to infer additional information from encrypted dataset, requested set queries and corresponding responses.

2.3 Problem statement

Let a dataset $\mathcal{DS} = (l_i, S_i)_{i=1}^{n}$ is a list of label/set pairs, where $l_i \in \{0,1\}^{\lambda}$ and $S_i = (e_1^i, \cdots, e_{k_i}^i)$. Also, let a set query is a triple $q = (op, l_k, l_p)$ in which op is one of the set operations mentioned below, and l_k and l_p are two arbitrary set labels:

- $\cap$: the intersection of two sets,
- $\cup$: the union of two sets,
- $\setminus$: the difference between two sets.

In such a setting and based on the scenario and threat models mentioned above, our problem is to design cryptographic primitives for the set operation functionality. These primitives enable the user to securely outsource his dataset to a cloud service provider and later privately issues queries in the form of common set operations.

3 Preliminaries

In this section, we introduce the notations used throughout the paper, as well as review the concepts and cryptographic primitives we will need in this paper. Readers familiar with these cryptographic primitives and concepts can safely skip Subsections 3.2 and 3.3, however we recommend studying the notations in Subsection 3.1.

3.1 Notations

Throughout the paper, we consider the symbol λ to denote the security parameter. We denote efficient probabilistic algorithms with PPT, which is an abbreviation for **P**robabilistic **P**olynomial **T**ime, and it means that the time complexity of the algorithm is polynomial in terms of the security parameter. We say that a function such as ε is negligible, if for every sufficiently large security parameter λ, the function ε is smaller than the inverse of any polynomial in terms of λ. Suppose that S is a finite set, we use $e \leftarrow S$ to indicate that the element e is uniformly chosen from the set S, and $\mid S \mid$ to denote the cardinal number S. Let A is a probabilistic algorithm and x is a candidate input for it, we use $y \leftarrow A(x)$ to denote that y is the output of A on input x. For the security experiments defined in this paper, we denote the output of an experiment by the experiment name itself. Also, we use the notation $[n]$ to denote the set $\{1, 2, \cdots, n\}$, where n is a positive integer number. The relation "$A \implies B$" indicates that the notion A implies the notion B. The relation "$A \not\Rightarrow B$" stands for a separation between the notions A and B.

3.2 Cryptographic concepts

In this subsection, we recall some of the basic cryptographic concepts that we use in our security proof. The readers familiar with these concepts can safely skip this subsection.

Computational indistinguishability. Let X_λ, Y_λ be distributions over $\{0,1\}^{l(\lambda)}$ for some polynomial $l(\lambda)$. We say that the families $\{X_\lambda\}_\lambda$ and $\{Y_\lambda\}_\lambda$ are computationally indistinguishable, and write $X_\lambda \approx_c Y_\lambda$, if for every PPT distinguisher $\mathcal{D}$, there exists a negligible function ε such that:

$$| \Pr[t \leftarrow X_\lambda : \mathcal{D}(t) = 1] - [t \leftarrow Y_\lambda : \mathcal{D}(t) = 1] | \leq \varepsilon(\lambda).$$

Closure under efficient operations. For a pair of distributions X_λ and Y_λ, if $X_\lambda \approx_c Y_\lambda$ then for any PPT algorithms M, it holds that $M(X_\lambda) \approx_c M(Y_\lambda)$. This is known as the closure under efficient operations.

Hybrid argument. Let $X_1, X_2, \cdots, X_m$ be a sequence of probability distributions. Assume that the distinguisher $\mathcal{D}$ can distinguish between X_1 and X_m with advantage ε. Then, there exists some $i \in \{1, \cdots, m-1\}$ such that the distinguisher $\mathcal{D}$ can distinguish X_i and X_{i+1} with advantage $\frac{\varepsilon}{m}$. This is known as the hybrid argument.

3.3 Cryptographic primitives

In this subsection, we recall some of the basic cryptographic primitives that we use in our construction. The readers familiar with these primitives and concepts can safely skip this subsection. However, we encourage the reader to take a look at the formal definitions of Tset and Adjoin schemes.

SKE scheme. The symmetric-key encryption (SKE) scheme is a cryptographic primitive that enable two parties to communicate secretly in the presence of an eavesdropper who can monitor all communication between them. In the following, the formal definition of SKE schemes is given.

Definition 1 (SKE syntax [11]) A symmetric-key encryption scheme is a triple of PPT algorithms SKE = (KeyGen, Enc, Dec) such that:

- $k \leftarrow \mathsf{KeyGen}(1^\lambda)$: is a probabilistic key generation algorithm that takes as input a security parameter λ, and returns a secret key k.
- $c \leftarrow \mathsf{Enc}_k(m)$: is a probabilistic (or deterministic) encryption algorithm that takes as input a secret key k and a message m, and returns as output a ciphertext c.
- $m \leftarrow \mathsf{Dec}_k(c)$: is a deterministic decryption algorithm that takes as input a secret key k and a ciphertext c, and returns a message m.

The correctness condition of the SKE scheme guarantees that decryption of any encrypted plaintext leads us to the same plaintext.

The security goal of a SKE is to keep the plaintext hidden from an eavesdropper who can monitor the communication channel, and also has access to the encryption machine for a short period of time. In the literature, the SKE scheme that meets such a security goal is referred to as CPA-secure.

Tset scheme. The tuple set (Tset) scheme, first proposed by Cash et. al in [3], is a cryptographic primitive that enables a user to securely store a hashed

form of a datatable and to generate a token for each row. The token can then be used to retrieve the corresponding row publicly from the hashed datatable. In the following, the formal definition of Tset schemes is given.

Definition 2 (Tset syntax [3]) A tuple set scheme is a collection of four PPT algorithms $\mathsf{Tset} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Token}, \mathsf{Retrive})$ such that:

- $(K, \mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda)$: is a probabilistic setup algorithm that takes as input a security parameter λ, and returns a secret key K, and an empty datatable $\mathcal{T}$.
- $\mathcal{T} \leftarrow \mathsf{Hash}_K\big(\mathcal{T}, (e, l)\big)$: is a hash algorithm that takes as input a secret key K, a datatable $\mathcal{T}$, and an element/label pair (e, l) that it determines an element with value e from a set with label l, and updates the hashed datatable $\mathcal{T}$.
- $rt \leftarrow \mathsf{Token}_K(l)$: is a token generation algorithm that takes as input a secret key K and a retrieval query l, and returns a retrieval token rt.
- $R \leftarrow \mathsf{Retrive}(\mathcal{T}, rt)$: is a deterministic algorithm that takes as input a hashed datatable $\mathcal{T}$ and a retrieval token rt, and returns a plaindata row R.

The correctness condition of the Tset scheme guarantees that no efficient adversary is able to provide datatable $\mathcal{T}$ and retrieval query l in such a way that algorithm Retrieve returns a result other than the expected result.

The security goal in a Tset scheme is to hide the elements of a set in a dataset until a set is retrieved. Cash et al. have investigated and defined the information leakages of such schemes in [3]. In this regard, they consider an experiment in which the adversary and the challenger interact adaptively through algorithms Hash and Token. In such an experiment, we expect that the adversary will not get information beyond the allowed leakages.

Adjoin **scheme.** The adjustable join (Adjoin) scheme, first proposed by Popa et al. in [12], is a cryptographic primitive that supports secure join queries for pairs of column labels on an encrypted database. In the following, the formal definition of Adjoin schemes is given.

Definition 3 (Adjoin syntax [4]) An adjustable join scheme is a collection of four PPT algorithms $\mathsf{Adjoin} = (\mathsf{KeyGen}, \mathsf{Encod}, \mathsf{Token}, \mathsf{Adjust})$ such that:

- $(Param, K) \leftarrow \mathsf{KeyGen}(1^\lambda)$: is a probabilistic key generation algorithm that takes as input a security parameter λ, and returns a secret key K and a set of the public parameters $Param$.
- $\widetilde{e} \leftarrow \mathsf{Encod}_K(e, l)$: is a deterministic encoding algorithm that takes as input a secret key K, an element e and a column label l, and outputs an encoded-element $\widetilde{e}$.
- $(at_k, at_p) \leftarrow \mathsf{Token}_K(l_k, l_p)$: is a token generation algorithm that takes as input a secret key K and two column labels (l_k, l_p), and returns a pair (at_k, at_p) of adjustment tokens.

- $ae \leftarrow \mathsf{Adjust}_{Param}(\widetilde{e}, at)$: is a deterministic algorithm that takes as input the public parameters $Param$, an encoded-element $\widetilde{e}$ and an adjustment token at, and outputs an adjusted encoded-element ae.

The correctness condition of an Adjoin schemes guarantees that no efficient adversary is able to provide two column labels l_k, l_p and two elements $e_1 \neq e_2$ such that the output of algorithm Adjust shows that their adjusted encoded-elements are the same.

The security goal in an Adjoin scheme is to not reveal any useful information about the join queries and database elements. Mironov et al. have defined the information leakages of such schemes in [8]. Also, they have proposed several security notions for the Adjoin schemes and investigated the relationship between them. In particular, one of the security notions that they presented and we use is the aIND security notion. This security notion considers an experiment in which the adversary and the challenger interact adaptively through algorithms Encod and Token. In such an experiment, we expect that the adversary will not get information beyond the allowed leakages.

4 PSO schemes

In this section, we first review the syntax notion of the PSO schemes. Next, we study the only security notion proposed for PSO schemes, which is called naSIM security. Finally, we describe a generic PSO construction that is presented in [5] and satisfies the mentioned syntax and security notions. It should be noted that the mentioned generic construction is the foundation of the construction that we present in this paper, by determining its building blocks.

4.1 PSO syntax

A private set operation (PSO) scheme is a cryptographic primitive that is used to encrypt a dataset, and to generate a pair of tokens to compute a set operation on any two chosen sets. In the following, we first explain how to use a PSO scheme, and then formally define its syntax.

A user who wants to securely outsource his dataset to an external server, in the first step runs the algorithm Setup to obtain a set of public parameters $Param$, a secret key K and an empty encrypted-dataset $\widetilde{\mathcal{DS}}$, and then sends the public parameters $Param$ to the server. In the second step, for every element e relative to any set with label l, the user calls the encryption algorithm Enc and receives an updated version of the encrypted-dataset. After the encrypted-dataset $\widetilde{\mathcal{DS}}$ is gradually completed by calling the encryption algorithm Enc, the user outsources $\widetilde{\mathcal{DS}}$ to the server. Later, when the user wants to send a set query q to the server, he computes a search token sot using a token generation algorithm denoted by Token. Upon receiving search token sot, the server computes an encrypted result set using the set operation algorithm denoted by Operate, and sends it to the user. Finally, the user extract the

result set from the encrypted result set using a decryption algorithm denoted by Retrieve. In the following, the formal definition of this primitive is given.

Definition 4 (PSO syntax [5]) A private set operation scheme is a collection of five PPT algorithms $\mathsf{PSO} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Operate}, \mathsf{Retrieve})$ such that:

- $(Param, K, \widetilde{\mathcal{DS}}) \leftarrow \mathsf{Setup}(1^\lambda)$: is a probabilistic setup algorithm that takes as input a security parameter λ, and returns the public parameters $Param$, a secret key K, and an empty encrypted-dataset $\widetilde{\mathcal{DS}}$.
- $\widetilde{\mathcal{DS}} \leftarrow \mathsf{Enc}_K\big(\widetilde{\mathcal{DS}}, (e, l)\big)$: is an encryption algorithm that takes as input a secret key K, an encrypted-dataset $\widetilde{\mathcal{DS}}$, and an element/label pair (e, l) that it determines an element with value e from a set with label l, and updates the encrypted-dataset $\widetilde{\mathcal{DS}}$.
- $sot \leftarrow \mathsf{Token}_K(q)$: is a token generation algorithm that takes as input a secret key K and a set query q, and returns a set operation token sot.
- $\widetilde{S} \leftarrow \mathsf{Operate}_{Param}(\widetilde{\mathcal{DS}}, sot)$: is a deterministic algorithm that takes as input the public parameters $Param$, an encrypted-dataset $\widetilde{\mathcal{DS}}$ and a set operation token sot, and returns an encrypted-set result $\widetilde{S}$.
- $S \leftarrow \mathsf{Retrieve}_K(\widetilde{S})$: is a deterministic algorithm that takes as input a secret key K and an encrypted-set $\widetilde{S}$, and returns a plain-set S.

Correctness. The scheme is said to be correct, if for any security parameter λ, any dataset $\mathcal{DS} = \big((l_i, s_i)\big)_{i\in[n]}$ where $l_i \in \{0,1\}^\lambda$ and $s_i = (e^i_1, \cdots, e^i_{n_i}) \in \{0,1\}^{\lambda n_i}$, and any set operation query $q = (op, l_k, l_p)$, it holds that:

$$\mathsf{Adv}^{\mathsf{Cor}}_{\mathsf{PSO}}(\lambda) = \Pr\left[\begin{array}{l}(Param, K, \widetilde{\mathcal{DS}}) \leftarrow \mathsf{Setup}(1^\lambda);\\ \forall i \in [m],\ j \in [n_i] \qquad \widetilde{\mathcal{DS}} \leftarrow \mathsf{Enc}_K\big(\widetilde{\mathcal{DS}}, (e, l)\big);\\ sot \leftarrow \mathsf{Token}_K(q) :\\ \mathsf{Retrieve}_K\big(\mathsf{Operate}_{Param}(\widetilde{\mathcal{DS}}, sot)\big) = (s_k \quad op \quad s_p)\end{array}\right] = 1. \tag{1}$$

4.2 Non-adaptive simulation-based security notion

In this subsection, we review a simulation-based security notion against non-adaptive adversary for PSO schemes, which is proposed in [5] and is called naSIM. It should be noted that the this security notion is the only security notion proposed for the PSO schemes.

Since that providing a formal definition for the above security notion requires the introduction of leakage function, we first review the proposed reasonable leakages for PSO schemes [5].

PSO leakage function. For a dataset $\mathcal{DS}$ and a list of set queries $\mathcal{Q}$, the minimal leakage function for the PSO schemes are defined as follows:

$$(\mathrm{Dim}, \mathsf{SP}, \mathsf{OPP}, \mathsf{RP}, \mathsf{DP}) \leftarrow \mathsf{Leak}(\mathcal{DS}, \mathcal{Q})$$

where the leakage profile includes:

- **Dimension (Dim).** It is the dataset dimensions that determines the total number of sets and the cardinality of each set,
- **Search pattern (SP).** It specifies the repetition of retrieved sets in different queries,
- **Operation pattern (OPP).** For each query issued, it determines the operation pattern,
- **Result pattern (RP).** For each query issued, it shows the result of the set query,
- **Duplication pattern (DP).** For each set in the dataset, it determines the locations with the same content.

The formal definition and details of each of the above components are given in [5], and we refer the interested readers to it.

naSIM security notion. This security notion considers an adversary $\mathcal{A}$, a simulator $\mathcal{S}$ and two experiments: *Real-world* and *Ideal-world*. In the *Real-world* experiment, the adversary $\mathcal{A}$ interacts with the scheme in the following non-adaptive manner: it obtains the public parameters of the scheme, chooses a dataset $\mathcal{DS}$ and a query list $\mathcal{Q}$, and finally obtains an encrypted dataset and a list of set operation tokens. In the *Ideal-world* experiment, the simulator has to produce a view that is indistinguishable from the *Real-world* when given only the leakage profile $(\mathsf{Dim}, \mathsf{SP}, \mathsf{OPP}, \mathsf{RP}, \mathsf{DP}) \leftarrow \mathsf{Leak}(\mathcal{DS}, \mathcal{Q})$ (i.e. without being given the dataset $\mathcal{DS}$ and the query list $\mathcal{Q}$). In fact, this security notion guarantees that no information beyond the leakage given to the simulator is revealed. If additional leakage occurs, the adversary can obtain two distinct views of the real and ideal worlds, and as a result, the security of the scheme is violated. Therefore, it is essential to choose a reasonable and standard leakage, which is determined for PSO schemes in paper [5] (see the leak function Leak above). The formal definition of this security notion is as follows.

Definition 5 (naSIM security [5]) A private set operation scheme $\mathsf{PSO} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Operate}, \mathsf{Retrieve})$ is Leak-semantically-secure against non-adaptive adversaries, if for every PPT adversary $\mathcal{A}$ there exist a PPT simulator $\mathcal{S}$ and a negligible function ε such that

$$| \Pr[\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A},\mathcal{S}}(\lambda) = 1] | \leq \varepsilon(\lambda),$$

where the experiments $\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A},\mathcal{S}}(\lambda)$ are defined as follows:
Real world $(\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda))$**:**

1. **Setup phase:** The challenger runs $(Param, K, \widetilde{\mathcal{DS}}) \leftarrow \mathsf{Setup}(1^\lambda)$, and returns the public parameters $Param$ to the adversary $\mathcal{A}$. Also, the challenger defines an empty list SOT.
2. **Challenge phase:** The adversary $\mathcal{A}$ takes the security parameter λ, chooses a dataset $\mathcal{DS}$ and a list of set queries $\mathcal{Q}$, and sends $\mathcal{DS}$ and $\mathcal{Q}$ to the challenger.

3. **Response phase:** For every $(e, l) \in \mathcal{DS}$, the challenger updates $\widetilde{\mathcal{DS}}$ using $\widetilde{\mathcal{DS}} \leftarrow \mathsf{Enc}_K\big(\widetilde{\mathcal{DS}}, (e, l)\big)$. Also, for every $q \in \mathcal{Q}$, he runs $sot \leftarrow \mathsf{Token}_K(q)$ and adds it to SOT. Finally, he returns $\widetilde{\mathcal{DS}}$ and SOT to the adversary $\mathcal{A}$.
4. **Output phase:** The adversary $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the output of the experiment.

Ideal world ($\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A},\mathcal{S}}(\lambda)$)**:**

1. **Setup phase:** The simulator runs $(Param^*, K^*, \widetilde{\mathcal{DS}}^*) \leftarrow \mathsf{Setup}(1^\lambda)$, and returns the public parameters $Param^*$ to the adversary $\mathcal{A}$. Also, the simulator defines an empty list SOT^*.
2. **Challenge phase:** The adversary $\mathcal{A}$ takes the security parameter λ, chooses a dataset $\mathcal{DS}$ and a list of set queries $\mathcal{Q}$, and sends $\mathcal{DS}$ and $\mathcal{Q}$ to the challenger.
3. **Response phase:** The challenger computes the leakage profile $L \leftarrow \mathsf{Leak}(\mathcal{DS}, \mathcal{Q})$, and gives L as input to the simulator. The simulator $\mathcal{S}$ produces an encrypted-dataset $\widetilde{\mathcal{DS}}^*$ and a list of set operation tokens SOT^*, which are then given to the adversary $\mathcal{A}$.
4. **Output phase:** The adversary $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the output of the experiment.

4.3 Generic PSO construction

In this subsection, we review the generic PSO construction proposed in [5], which is the basis of the new construction that we propose in this paper. This construction uses: a **S**ymmetric-**K**ey **E**ncryption (SKE) scheme to encrypt the elements of each set in the dataset, an **A**djustable **Join** (Adjoin) scheme to generate the encoded-element for every single element of every set in the dataset, as well as a token for the equality test of elements in two sets, and a **T**uple **S**et (Tset) scheme to distribute the set elements uniformly across a datatable and to hide the occurrence pattern of the elements in the dataset. In the following, each of the algorithms of the generic PSO construction is described.

Algorithm 1 shows the key generation process. As it can be seen, the key generation algorithm Adjoin.KeyGen produces key k_A to generate the adjustment encoded-elements and the adjustment tokens, as well as the public parameters $Param$ (line 1). In the following lines, the key k_E is generated by the SKE.KeyGen algorithm to encrypt the set elements, and the key k_T is generated by the Tset.Setup algorithm to distribute the set elements uniformly across a dataset (lines 2,3). In this algorithm, a data structure, denoted by $\widetilde{\mathcal{DS}}$, is also configured to hold the encrypted-dataset.

Algorithm 1 $(Param, K, \widetilde{\mathcal{DS}}) \leftarrow \mathsf{Setup}(1^\lambda)$

1: $(Param, k_A) \leftarrow \mathsf{Adjoin.KeyGen}(1^\lambda)$
2: $k_E \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$

3: $(k_T, \widetilde{\mathcal{DS}}) \leftarrow \mathsf{Tset.Setup}(1^\lambda)$
4: Output $\left(Param, K = (k_E, k_A, k_T), \widetilde{\mathcal{DS}}\right)$

Algorithm 2 shows the process of updating the encrypted-dataset $\widetilde{\mathcal{DS}}$ based on the pair (e, l). In this process, first an encoded-element $\widetilde{e}$ is generated by Adjoin.Encod algorithm, then it is encrypted by SKE.Enc algorithm and finally placed in an appropriate location in encrypted-dataset $\widetilde{\mathcal{DS}}$ by Tset.Hash algorithm.

Algorithm 2 $\widetilde{\mathcal{DS}} \leftarrow \mathsf{Enc}_K\left(\widetilde{\mathcal{DS}}, (e, l)\right)$

1: Parse K as (k_E, k_A, k_T)
2: $\widetilde{e} \leftarrow \mathsf{Adjoin.Encod}_{k_A}(e, l)$
3: $c = \mathrm{SKE.Enc}_{k_E}(e)$
4: $\mathsf{Tset.Hash}_K\left(\widetilde{\mathcal{DS}}, \left((c, \widetilde{e}), l\right)\right)$
5: Output $\widetilde{\mathcal{DS}}$

Algorithm 3 shows the token generation process for a set query. As it can be seen, to retrieve the desired sets, we first run Tset.Token algorithm and obtain the retrieval tokens. Then, using Adjoin.Token algorithm , we generate adjustment tokens to test the equality of the retrieved elements.

Algorithm 3 $sot \leftarrow \mathsf{Token}_K(q)$

1: Parse K as (k_E, k_A, k_T)
2: Parse q as (op, l_k, l_p)
3: $rt_k \leftarrow \mathsf{Tset.Token}_{k_T}(l_k)$
4: $rt_p \leftarrow \mathsf{Tset.Token}_{k_T}(l_p)$
5: $(at_k, at_p) \leftarrow \mathsf{Adjoin.Token}_K(l_k, l_p)$
6: Output $sot = (op, rt_k, rt_p, at_k, at_p)$

Algorithm 4 shows the process of computing a set operation on an outsourced dataset. In this process and in the first step, the queried sets are fetched using retrieval tokens. Then, using the adjustment tokens, it computes the adjusted encoded-elements of each element of the fetched sets. The above computation reduces the set operations on encrypted-dataset to the set operations on the plain-dataset.

Algorithm 4 $\widetilde{S} \leftarrow \mathsf{Operate}_{Param}(\widetilde{\mathcal{DS}}, sot)$

1: Parse sot as $(op, rt_k, rt_p, at_k, at_p)$
2: $R_k \leftarrow \mathsf{Tset.Retrive}(\widetilde{\mathcal{DS}}, rt_k)$
3: $R_p \leftarrow \mathsf{Tset.Retrive}(\widetilde{\mathcal{DS}}, rt_p)$

Initialize the empty lists $\widetilde{R_k}$, $\widetilde{R_p}$, $\widetilde{S}_I$ and $\widetilde{S}$.
for every $(c, \widetilde{e}) \in R_k$ **do**
    $ae = \mathsf{Adjoin.Adjust}_{Param}(\widetilde{e}, at_k)$
    Append (c, ae) to $\widetilde{R_k}$
for every $(c, \widetilde{e}) \in R_p$ **do**
    $ae = \mathsf{Adjoin.Adjust}_{Param}(\widetilde{e}, at_p)$
    Append (c, ae) to $\widetilde{R_p}$
$\widetilde{S}_I = \big((c, ae) \mid (c, ae) \in \widetilde{R_k}$ and $\exists c'$ s.t. $(c', ae) \in \widetilde{R_p}\big)$
if $(op == \cap)$ **then**
    $\widetilde{S} = \big(c \mid \exists (c, ae) \in \widetilde{S}_I\big)$
if $(op == \cup)$ **then**
    $\widetilde{S} = \big(c \mid [\exists (c, ae) \in \widetilde{R_p}]$ or $[\exists (c, ae)$ s.t. $(c, ae) \in \widetilde{R_k}$ and $(c, ae) \notin \widetilde{S}_I]\big)$
if $(op == \setminus)$ **then**
    $\widetilde{S} = \big(c \mid \exists (c, ae)$ s.t. $(c, ae) \in \widetilde{R_k}$ and $(c, ae) \notin \widetilde{S}_I\big)$
Output $\widetilde{S}$

Algorithm 5 shows the process of extracting the plain result set from the encrypted result set returned by Operate algorithm. More precisely, each of the encrypted-elements is converted into an plain-element by the SKE.Dec algorithm and added to the plain result set.

Algorithm 5 $S \leftarrow \mathsf{Retrieve}_K(\widetilde{S})$

Parse K as (k_E, k_A, k_T)
Initialize S as an empty list
for every $c \in \widetilde{S}$ **do**
    Append $\mathsf{SKE.Dec}_{k_E}(c)$ to S
Output S

In paper [5], and in the form of the following theorem, Rafiee et al. showed that if a PSO construction is proposed with the above approach, its correctness condition is guaranteed by the correctness condition of constructions SKE, Adjoin and Tset.

Theorem 1 (PSO correctness condition [5]) *If* Tset, Adjoin *and* SKE *used in the above* PSO *construction satisfy the correctness condition of their schemes, then our* PSO *construction is correct.*

5 A new security notion for PSO schemes

In this section, we first propose a indistinguishability-based security notion against adaptive adversary for PSO schemes, which is called aIND security. It should be noted that this security notion has been defined and developed based on security notions raised in the literature. In the rest of this section, we also investigate the implication and separation of this new security notion with naSIM (see Definition 5), and we show that our security notion is strictly stronger than naSIM.

5.1 Adaptive indistinguishability-based security notion

In this subsection, we redefine the indistinguishability-based security notion against adaptive adversary of SSE schemes, which is proposed in [13], for PSO schemes.

The experiment defined for this security notion considers an adversary who receives the public parameters of the PSO scheme at the beginning of the experiment. Then, he can adaptively issue a pair of challenge elements e_0, e_1 for any arbitrary label l, and gets an updated version of the dataset based on some fixed bit $b \in \{0,1\}$. As soon as the adversary issues such queries, the challenge elements are appropriately added to the datasets $\mathcal{DS}_0$ and $\mathcal{DS}_1$, and they are gradually filled. In addition, the adversary can adaptively issue a set query, and obtain a set operation token. Let $\mathcal{Q}$ be the list of set queries that the adversary has obtained their tokens at a given point of time. Such an adversary is called a valid adversary if, after having made each query (Enc or Token), it holds that $\mathsf{Leak}(\mathcal{DS}, \mathcal{Q}) = \mathsf{Leak}(\mathcal{DS}, \mathcal{Q})$. It should be noted that for the simplicity of the game, and instead of including constraints in the game that prevent the adversary from winning obviously, we used the concept of the valid adversary based on the leakage function defined in Subsection 4.2. The (aIND) security notion requires that such a valid adversary is not able to distinguish between the experiment in which $b = 0$ and the experiment in which $b = 1$, with a non-negligible advantage. The formal definition of this security notion is as follows.

Definition 6 (Leak-aIND security) An PSO scheme such as PSO = (Setup, Enc, Token, Operate, Retrieve) is called adaptively indistinguishable-based secure with respect to the leakage function Leak (abbreviated Leak-aIND-secure) if for every PPT valid adversary $\mathcal{A}$, there exists a negligible function ε such that

$$| \Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda, b) = b] | \leq \frac{1}{2} + \varepsilon(\lambda),$$

where for each $b \in \{0,1\}$, the experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda, b)$ is defined as follows:

1. **Setup phase:** The challenger runs $(Param, K, \widetilde{\mathcal{DS}}) \leftarrow \mathsf{Setup}(1^\lambda)$, and returns the public parameters $Param$ to the adversary $\mathcal{A}$. In addition, two empty dataset $\mathcal{DS}_0, \mathcal{DS}_1$ and an empty query list $\mathcal{Q}$ are initialized.
2. **Challenge phase:** The adversary $\mathcal{A}$ may adaptively issue $\mathsf{Enc}(\cdot,\cdot,\cdot)$ and $\mathsf{Token}(\cdot)$ queries, which are defined as follows:

- $\mathsf{Enc}(e_0, e_1, l)$: The pairs (e_0, l) and (e_1, l) are inserted into the dataset $\mathcal{DS}_0$ and $\mathcal{DS}_1$, respectively. Then, the challenger sends an updated dataset $\widetilde{\mathcal{DS}} \leftarrow \mathsf{Enc}_K\big(\widetilde{\mathcal{DS}}, (e_b, l)\big)$ to the adversary $\mathcal{A}$.
- $\mathsf{Token}(q)$: The query $q = (op, l_k, l_p)$ is inserted into the list $\mathcal{Q}$, and the challenger sends the set operation token $sot \leftarrow \mathsf{Token}_K(q)$ to the adversary $\mathcal{A}$.

It is required that, after issuing each query, the equality $\mathsf{Leak}(\mathcal{DS}_0, \mathcal{Q}) = \mathsf{Leak}(\mathcal{DS}_1, \mathcal{Q})$ holds.

3. **Output phase:** The adversary $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the output of the experiment.

5.2 Relations between naSIM and aIND

In this subsection, we investigate the implications and separations between the two security notions naSIM and aIND. More precisely, we show the relations "naSIM $\Longrightarrow$ aIND" and "aIND $\not\Longrightarrow$ naSIM", and conclude that the security notion aIND is strictly stronger than the security notion naSIM. To prove the these claims, we need to define two concepts: *History* and *Non-singular history*, which are given below.

A history actually contains all the information exchanged in the interaction between the adversary and the challenger in experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda, b)$. The following definition formally states this concept.

Definition 7 (History) A tuple $H = (\mathcal{DS}, Q)$ is called a l-query history, where $\mathcal{DS}$ represents a dataset and Q represents a vector of l set operations ($Q = (q_1, \cdots, q_l)$).

It should be noted that for the security of aIND to be meaningful, it is essential that for every history H, there must be a different history H' such that $H' \neq H$. If there is no such history H', it means that:

- or the leakage function reveals all the information, in which case, based on the relation $\mathsf{Leak}(H) = \mathsf{Leak}(H')$, we have only one history $H = H'$,
- or the leakage function does not reveal all the information and at the same time we do not have a secondary history H', in which case the adversary has no chance of winning and the designed game is meaningless.

This basic necessity is known in the literature as non-singular history and its formal definition is given below.

Definition 8 (Non-singular history [13]) We say that a history H is non-singular if:

- there exists at least one history $H' \neq H$ such that $\mathsf{Leak}(H) = \mathsf{Leak}(H')$,
- such a history can be found in polynomial-time given $\mathsf{Leak}(H)$.

In the following, the security relations between each of the aforementioned notions are provided.

Proposition 2 (aIND $\implies$ naSIM) *Let* Leak *be the leakage functions defined in Section 4.2. If an* PSO *scheme is* Leak-aIND*-secure, then it is* Leak-naSIM*-secure too.*

Proof 1 To prove this claim, we use the proof by contradiction. In fact, we need to show that if a PSO scheme does not have Leak-naSIM security , then it does not cover Leak-aIND security. More precisely, we need to show that if there exists a PPT adversary $\mathcal{A}$ such that for every PPT simulator $\mathcal{S}$, it is able to distinguish between the outputs $\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A},\mathcal{S}}(\lambda)$ with a significant advantage $\mu(\lambda)$, i.e.,

$$| \Pr[\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A},\mathcal{S}}(\lambda) = 1] | > \mu(\lambda),$$

then there exists a PPT adversary $\mathcal{B}$ that succeeds in $\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda, b)$ with overwhelming probability, i.e.,

$$| \Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda, b) = b] | > \frac{1}{2} + \mu(\lambda).$$

Let $H_0 = (\mathcal{DS}, Q)$ be the output of $\mathcal{A}(\lambda)$ and recall that H_0 is non-singular. Therefore, there exists at least one history $H_1 \neq H_0$ such that $\mathsf{Leak}(H_0) = \mathsf{Leak}(H_1)$, and also it can be found efficiently. Without loss of generality, we assume that the adversary $\mathcal{A}$ outputs 0 when given the output of the experiment $\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda)$. In such a situation, we construct the adversary $\mathcal{B}$ who acts as follows:

- In the first step, the adversary $\mathcal{B}$ generates the history H_1 using $\mathsf{Leak}(H_0)$,
- In the second step, the adversary $\mathcal{B}$, based on histories H_0 and H_1, issues adaptively queries $\mathsf{Enc}_K(e_0, e_1, l)$ and $\mathsf{Token}_K(q)$, and constructs an encrypted-dataset $\widetilde{\mathcal{DS}}$ and a list of set operation tokens SOT, which are then given to the adversary $\mathcal{A}$.
- Finally, the adversary $\mathcal{B}$ outputs everything that adversary $\mathcal{A}$ returns.

In the following, we only need to show that the probability of the attacker's success is significant. Since b is chosen uniformly at random, we have:

$$\begin{aligned}\Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{B}}(\lambda, b) = 0] = \frac{1}{2}\big(\Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{B}}(\lambda, b) = 0 \mid b = 0]\big) \\ + \frac{1}{2}\big(\Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{B}}(\lambda, b) = 0 \mid b = 1]\big)\end{aligned} \tag{2}$$

If $b = 0$ then adversary $\mathcal{B}$ succeeds if and only if the adversary $\mathcal{A}$ outputs 0. Therefore, we have:

$$\Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{B}}(\lambda, b) = 0 \mid b = 0] = \Pr[\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda) = 0] \tag{3}$$

On the other hand, if $b = 1$ then $\mathcal{B}$ succeeds if and only if the adversary $\mathcal{A}$ outputs 1. In this case, we have:

$$\Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{B}}(\lambda, b) = 0 \mid b = 1] = 1 - \Pr[\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A}}(\lambda) = 1] \tag{4}$$

By combining Equations 3 and 4 with Equation 2, we have:

$$\begin{aligned}\Pr[\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{B}}(\lambda,b)=0] &= \frac{1}{2}\Big(\Pr[\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda)=0]+1-\Pr[\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A}}(\lambda)=0]\Big)\\ &= \frac{1}{2}+\frac{1}{2}\Big(\mid \Pr[\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{PSO},\mathcal{A}}(\lambda)=0]-\Pr[\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{PSO},\mathcal{A}}(\lambda)=0]\mid\Big)\\ &\geq \frac{1}{2}+\mu(\lambda)\end{aligned}$$

where $\mu(\lambda)$ is a non-negligible function in λ, and where the inequality follows from our original assumption about the adversary $\mathcal{A}$. □

Proposition 3 (naSIM $\not\Longrightarrow$ aIND) *Let* Leak *be the leakage functions defined in Section 4.2. Then, the* Leak-naSIM *security does not necessarily imply the* Leak-aIND *security.*

Proof 2 Let $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Operate}, \mathsf{Retrieve})$ be some Leak-aIND-secure PSO scheme. We construct an PSO scheme $\widetilde{\Pi} = (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Token}}, \widetilde{\mathsf{Operate}}, \widetilde{\mathsf{Retrieve}})$ which is yet naSIM-secure, with the same leakage function Leak, however it is not Leak-aIND-secure. The scheme $\widetilde{\Pi}$ is constructed as follows:

- $\widetilde{\Pi}.\widetilde{\mathsf{Setup}}(1^\lambda)$**:** It runs $(Param, K, \widetilde{\mathcal{DS}}) \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$ and chooses a random element $e^* \in \{0,1\}^\lambda$, and outputs $(Param, \widetilde{K}, \widetilde{\mathcal{DS}})$, where $\widetilde{K} = (K, e^*)$.
- $\widetilde{\Pi}.\widetilde{\mathsf{Enc}}_{\widetilde{K}}(l, e)$**:** It updates the encrypted-dataset $\widetilde{\mathcal{DS}}$ as follows:

$$\widetilde{\mathcal{DS}} = \begin{cases} \text{Add } \Big(\Pi.\mathsf{Enc}_K\big(\widetilde{\mathcal{DS}}, (e,l)\big), e^*\Big) \text{ to } \widetilde{\mathcal{DS}} & e \neq e^* \\ \text{Add } \Big(\Pi.\mathsf{Enc}_K\big(\widetilde{\mathcal{DS}}, (e,l)\big), 0^\lambda\Big) \text{ to } \widetilde{\mathcal{DS}} & e = e^* \end{cases}.$$

It is easy to see that $\widetilde{\Pi}$ is still naSIM-secure without any change in the leakage function, since the adversary's advantage only increases by $\frac{1}{2^\lambda}$ due to his ability to guess e^*. It is also easy to construct a valid aIND-attacker for $\widetilde{\Pi}$ with advantage one, since the adversary learns e^* after issuing the first query. He will then use it as one of his challenge elements. □

6 An aIND-secure PSO construction

In this section, based on the generic PSO construction proposed in Subsection 4.3, we first introduce a new instantiation for PSO schemes. Then, we prove that the proposed scheme satisfies the aIND security notion.

Our instantiation. To cover security level aIND, we require cryptographic primitives with the following description:

- A symmetric-key encryption (SKE) scheme with CPA security. There are considerable amount of options for choosing a secure and efficient SKE. From the available choices, we chose a 128-bit AES symmetric-key encryption scheme in CTR mode.

- A tuple set (Tset) scheme with aIND security. Unlike SKE schemes, the choices for a secure and efficient Tset scheme are limited. In this regard, we use the Tset scheme proposed in [3] for our instantiation.
- A adjustable join scheme (Adjoin) scheme with aIND security. Similar to Tset schemes, the choices for a secure and efficient Adjoin scheme are also limited. For this purpose, we use the Adjoin scheme introduced in [4] for our instantiation.

Correctness of our instantiation. It is clear that based on Theorem 1 and also the correctness condition of schemes SKE, Tset and Adjoin, the above instantiation satisfies the correctness condition of PSO schemes.

aIND **security of our instantiation.** Next, using standard cryptographic concepts and techniques such as: Computational Indistinguishability and Hybrid Argument, we prove that the above instantiation provides the aIND security level.

Theorem 4 *If* SKE *is a CPA-secure symmetric encryption scheme, and* Tset *and* Adjoin *are* aIND*-secure with leakage function defined in Subsection 4.2, then the* PSO *construction proposed in Section 6 is* Leak-aIND*-secure.*

Proof 3 We use a hybrid argument in a sequence of experiments to show that our PSO construction is Leak-aIND-secure, i.e.,

$$\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,1).$$

In this regard, based on the value b, we consider the following two sequences of experiments:

- **Sequence 1 ($b=0$):** this sequence begins with experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0)$, is followed by several intermediate experiments, and ends with experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda)$, in which we show that:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda),$$

- **Sequence 2 ($b=1$):** this sequence begins with experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,1)$, is followed by several intermediate experiments, and ends with experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda)$, in which we show that:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,1) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda),$$

where $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda)$ is an experiment in which the output values of the algorithms used in the experiment are replaced by truly random values. Finally, using the mentioned indistinguishable sequences, we have:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda) \quad \text{and} \quad \mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,1),$$

and consequently, using the transitive property of indistinguishability notion, we have:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,1).$$

In the following, we only show sequence 1, and since sequence 2 can be argued with a similar method, we refrain from mentioning it. For every experiment in the sequence, we first explain it and then argue that it is computationally indistinguishable from its previous experiment.

- $\underline{\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{SKE},\mathcal{A}}(\lambda,0)}$: We modify $\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0)$ as follows and denotes it by $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{SKE},\mathcal{A}}(\lambda,0)$. We replace the encrypted element c with an encryption of 0^λ (i.e. line 3 in Algorithm 2 is replaced with $c = \mathrm{SKE}.\mathsf{Enc}_{k_E}(0^\lambda)$. By CPA security of the symmetric-key encryption scheme SKE, it holds that:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\mathsf{PSO},\mathcal{A}}(\lambda,0) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{SKE},\mathcal{A}}(\lambda,0).$$

- $\underline{\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Adjoin},\mathcal{A}}(\lambda,0)}$: We modify $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{SKE},\mathcal{A}}(\lambda,0)$ as follows and denote it by $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Adjoin},\mathcal{A}}(\lambda,0)$. We replace the encoded-element $\widetilde{e_b}$ with an encoded-element of $e^{'}$ such that the leakage remains the same (i.e. line 2 in Algorithm 2 is replaced with $\widetilde{e} \leftarrow \mathrm{aIND}.\mathsf{Encod}_{k_A}(e^{'}, l)$). By aIND security of the adjustable join scheme Adjoin, it holds that:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{SKE},\mathcal{A}}(\lambda,0) \approx_c expr^{\mathsf{aIND}}_{\$\mathsf{Adjoin},\mathcal{A}}(\lambda,0).$$

- $\underline{\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Tset},\mathcal{A}}(\lambda,0)}$: We modify $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Adjoin},\mathcal{A}}(\lambda,0)$ as follows and denote it by $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Tset},\mathcal{A}}(\lambda,0)$. We randomly place the pair $(\widetilde{e}, l)$ in the empty cells of the dataset $\widetilde{\mathcal{DS}}$ instead of $\mathsf{Tset}.\mathsf{Hash}_K\left(\widetilde{\mathcal{DS}}, (\widetilde{e}, l)\right)$ (line 4 in Algorithm 2). By aIND security of the tuple set scheme Tset, it holds that:

$$\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Adjoin},\mathcal{A}}(\lambda,0) \approx_c \mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Tset},\mathcal{A}}(\lambda,0).$$

- $\underline{\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda)}$: Since that $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Tset},\mathcal{A}}(\lambda,0)$ does not depend on the value b and also all the output values of the algorithms used in the experiment are replaced by truly random values, we define the experiment $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{rand},\mathcal{A}}(\lambda)$, which is the same as $\mathsf{Exp}^{\mathsf{aIND}}_{\$\mathsf{Tset},\mathcal{A}}(\lambda,0)$, except that it has only one input parameter instead of two inputs.

7 Performance analysis

In this section, we compare the performance of our PSO scheme with other existing PSO schemes. More precisely, we first describe the settings and assumptions of our implementation. Then, we present asymptotic and concrete performance analyses for the computational and storage overheads of our scheme. Finally, we list the results of the experimental evaluation in a summary form.

7.1 Settings and assumptions

In this subsection, we describe system specification that we employed to extract the experimental results. Also, we explain the desired dataset and candidate PSO schemes for evaluation.

The candidate PSO constructions. In addition to the PSO construction proposed in Section 6, we consider the PSO constructions in [5], which are the only constructions available for PSO schemes and are all based on the generic PSO construction. Table 2 summarizes the specification of the candidate PSO constructions in terms of the used cryptographic tools, the security properties, and the asymptotic efficiency comparison.

Remark 2 We know that comparing the performance of cryptographic schemes is fair when they cover the same security level. However, as stated earlier and it can also be seen in Table 2, the above fairness property does not hold for our performance comparison because our PSO scheme covers a much stronger security level. In such a situation, our goal of the performance analysis is only to find a suitable view of the overheads of different algorithms, and also to show that our scheme, despite covering a higher level of security, holds the performance at the level of other schemes.

Remember that all of our candidate PSO schemes have two main ingredients: an Adjoin scheme and a Tset scheme. In such a situation, it should be noted that all the Adjoin schemes used in the candidate schemes use bilinear maps, which has a moderate overhead compared to other cryptographic operations. Consequently, the discussed PSO schemes are not suitable for large-scale applications. However, in particular, our construction in this paper is the only PSO construction with aIND security level.

System specification. We have implemented all our experiments in Java on a MacBook Air with an Apple M1 8-core processor running Sonoma 14.6.1 with 8 GB of RAM. We have used Type-D curve (parameter d224) for the pairing setting of the jPBC library [14].

Dataset collection. A detailed study of the algorithms that build the generic PSO scheme at the coarse-grained level (Algorithms 1 to 5) as well as at the fine-grained level (blocks used in building PSO scheme, such as: SKE, Tset and Adjoin) shows that the distribution of the dataset values does not affect the complexity of the algorithms, and only its quantity is matters. Furthermore, since our goal of performance analysis, according to Remark 2, is simply to compare candidate schemes with each other, we set our evaluation on a small scale for sets of sizes 10 to 50.

7.2 Asymptotic comparison

As it can be seen in the asymptotic efficiency column in Table 2, the computational overheads of each of the encryption, token generation, and intersection algorithms are linear in terms of the number and size of the sets, based on

Table 2: Specification and asymptotic efficiency comparisons of the candidate PSO constructions

Construction	Cryptographic tools		Security properties		Asymptotic efficiency		
	Tset	Adjoin	Hardness assumption	Security level	Encryption time	Token time	Intersection time
PSO I	Tset [3]	Adjoin from [8]	matix-DDH [9]	Leak-naSIM	$(2tn)$ EXP	(4) EXP	$(4n)$ BM
PSO II		Adjoin from [5]	XDHV [5]	Leak-naSIM	(tn) EXP	(2) EXP	$(2n)$ BM
Our PSO		Adjoin from [4]	DDH1 [10]	Leak-aIND	$(2tn)$ EXP	(2) EXP	$(2n)$ BM

t: the total number of the sets in the dataset, n: the maximum size of each set in the dataset, *exp*: group exponentiation, *bm*: bilinear map.

the heavy operators used in them. More precisely, the encryption overheads of our scheme are double compared to the lowest overhead of other candidate schemes. However, the computational overheads of our scheme for token generation and intersection are equal to the corresponding minimal overheads of the candidate schemes. As a result, if we accept the initial overhead for outsourcing, our scheme for a static data model, where data is outsourced only once and queried frequently, is the best scheme among the candidate schemes. The reason for this is that it provides us with a much higher security level without imposing overhead for future queries.

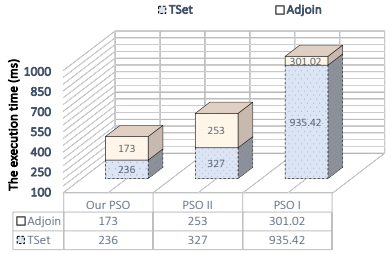
7.3 Concrete comparison

In the following, we concretely compare the candidate PSO schemes. In this regard, Figures 2 to 5 show our evaluation based on the time and space complexity of Algorithms Setup, Enc, Token and Operate.

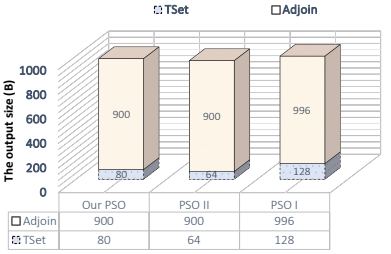
The setup algorithm. The execution time and output size of the setup algorithms are illustrated in Fig. 2a and 2b, respectively. As it can be seen in these figures, the imposed overheads of each component Tset and Adjoin have been shown separately. The experimental results of this section clearly show that component Tset occupies the majority of the execution time of the key generation algorithm, while the majority of the memory space is consumed by component Adjoin. As a summary, the storage overheads of all three candidate schemes are almost the same and do not have any particular advantage over each other. However, the execution time of the setup algorithm of our scheme reduces the computational overheads by approximately 30% and 67%, compared to schemes PSO II and PSO I, respectively.

The encryption algorithm. Figure 3 shows the evaluated experimental results for the encryption algorithms of the candidate PSO schemes. This section includes two evaluations: execution time and memory usage as follows:

- Figures 3a and 3b show the execution time evaluation on 10 to 50 elements. More precisely, Figure 3a shows the execution time of the encryption algorithm. As it can be seen, our scheme reduces the computational overheads by approximately 10% and 15%, compared to schemes PSO II and PSO I, respectively. Also, Figure 3b shows the execution time of the encryption algorithms related to the candidate schemes, separated into components

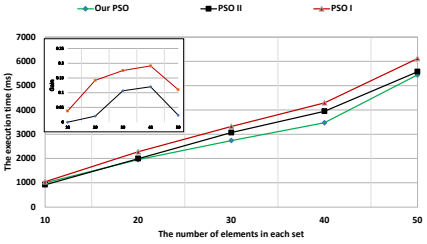


(a) Running time of the Setup algorithm.

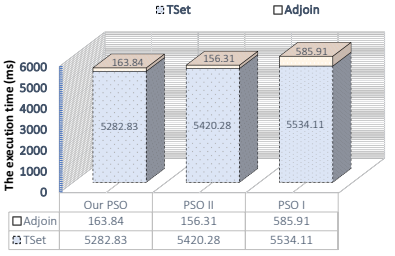


(b) Output size of the Setup algorithm.

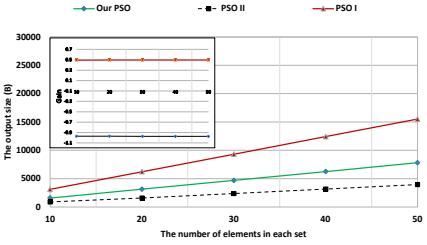
Fig. 2: Efficiency comparison of different key generation algorithms.



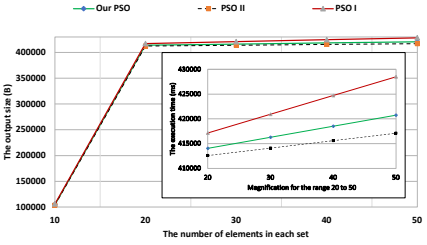
(a) Running time of the Enc algorithm.



(b) Running time of Enc algorithm with details (set size 50).



(c) Output size of the Encod algorithm.



(d) Output size of the Hash algorithm.

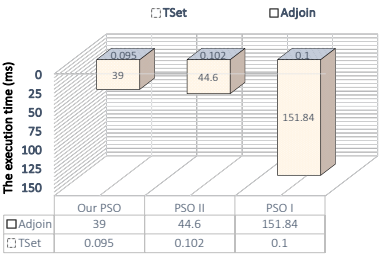
Fig. 3: Efficiency comparison of different encryption algorithms.

Tset and Adjoin, when the set size is 50. As it can be seen, component Adjoin occupies the majority of the execution time of the encryption algorithms.

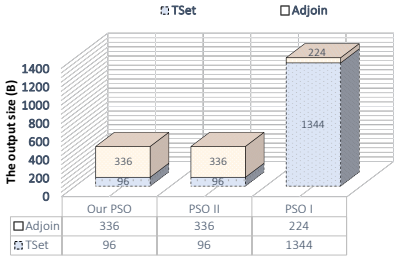
- Figures 3c and 3d show the memory usage evaluation on 10 to 50 elements. More precisely, Figure 3c shows the storage overheads obtained by the encryption algorithm in the encoding step (and before running the Tset.Hash algorithm and obtaining the encrypted-dataset). As it can be seen, our scheme, compared to PSO I, reduces storage overhead by approximately 50%, while compared to PSO II it increases overhead by approximately 90%.

However, Figure 3d shows that the storage overheads obtained from Algorithm Tset.Hash, which produces the final encrypted-dataset, are almost the same for all three candidate schemes and do not have a significant advantage over each other. This is due to the amount of redundancy imposed by Tset scheme to prevent some leakages.

The token generation algorithm. The execution time and output size of the token generation algorithms are illustrated in Figures 4a and 4b, respectively. As it can be seen in these figures, the imposed overheads of each component Tset and Adjoin have been shown separately. More precisely, Figure 4a shows that component Tset occupies the majority of the execution time of the token generation algorithm. Also, it shows that our scheme reduces the computational overheads by approximately 12.5% and 75%, compared to schemes PSO II and PSO I, respectively. Also, Figure 4b shows that our scheme, while having the same storage overhead as PSO II, reduces the storage overhead by approximately 72.5% compared to PSO I.



(a) Running time of the Token algorithm.



(b) Output size of the Token algorithm.

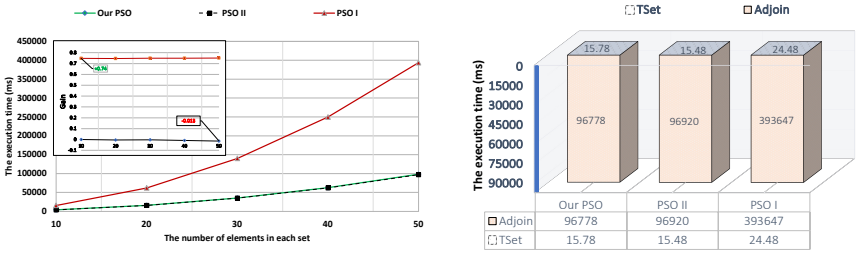
Fig. 4: Efficiency comparison of different token generation algorithms.

The set operation algorithm. Figures 5a and 5b show the intersection time on sets of sizes 10 to 50. In particular, Figure 5a shows that the execution time of our scheme is almost the same as PSO II, however our scheme reduces the computational overhead by approximately 74% compared to PSO I. Also, Figure 5b shows the execution time of the intersection algorithms related to the candidate schemes, separated into components Tset and Adjoin, when the set size is 50. As it can be seen, component Adjoin occupies the majority of the execution time of the intersection algorithms.

7.4 Summing up

The results of the above experimental evaluation can be summarized in the following three items:

- The results show that Adjoin schemes have the dominant execution time, and Tset schemes have the dominant memory usage. Therefore, to have a



(a) Running time of the Operate algorithm.

(b) Running time of Operate algorithm with details (set size 50).

Fig. 5: Efficiency comparison of different intersection algorithms.

more efficient PSO scheme based on the generic construction, we should try to provide Tset schemes with lower storage overheads and Adjoin schemes with lower computational overheads while still satisfying the desired security levels.

- In the literature, the trade-off between security and efficiency is always recognized as a principle, meaning that an increase in one usually leads to a decrease in the other, and vice versa. The results show that our scheme, despite covering a much higher security level, maintains efficiency at the level of other existing PSO schemes. In the literature, it is of great importance to propose cryptographic schemes that increase security while maintaining performance.
- The results show that our scheme, despite having a higher level of security and maintaining efficiency at the level of other existing PSO schemes, is still not suitable for large-scale applications due to its constituent components (Tset and Adjoin). However, our scheme is the only PSO scheme that has security level aIND and is available to organizations that require such a security level.

8 Conclusions and future works

In this paper, we proposed a new security notion (aIND) for the PSO schemes, which models security against adaptive adversaries. We showed that our security notion is stronger than the only existing security notion (naSIM) for the PSO schemes. We also proposed a new PSO construction that satisfied our security notion. In addition, we showed that our proposed scheme, despite having a higher level of security compared to other existing schemes, maintains efficiency at the level of existing schemes.

Future works can be developed in several directions. One branch is the development of the PSO schemes to support the multi-user model, where it is possible to issue set queries to multiple users according to predefined permissions [15, 16]. Another branch is the development of the PSO schemes to support verifiable computation, where the cloud server can act maliciously

rather than being honest-but-curious, and therefore the results returned from the server must be verified to ensure integrity [17, 18].

References

[1] Awais, M., Tahir, S., Khan, F., Tahir, H., Tahir, R., Latif, R., Umair, M.Y.: A novel searchable encryption scheme to reduce the access pattern leakage. Future Generation Computer Systems **133**, 338–350 (2022)

[2] Tahir, S., Steponkus, L., Ruj, S., Rajarajan, M., Sajjad, A.: A parallelized disjunctive query based searchable encryption scheme for big data. Future Generation Computer Systems **109**, 583–592 (2020)

[3] Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. Advances in Cryptology–CRYPTO 2013 **8042**, 353–373 (2013)

[4] Rafiee, M.: Multi-adjustable join schemes with adaptive indistinguishably security. IEEE Transactions on Dependable and Secure Computing **21**(4), 4024–4034 (2023)

[5] Rafiee, M., Khazaei, S.: Private set operations over encrypted cloud dataset and applications. The Computer Journal **64**(8), 1145–1162 (2021)

[6] Lv, S., Wei, Y., Jia, J., Li, X., Li, T., Liu, Z., Chen, X., Guo, L.: New approach for efficient malicious multiparty private set intersection. Information Sciences **678**, 120995 (2024)

[7] Qiao, X., Youwen, Z., WANG, X.J.: Locally differentially private distributed algorithms for set intersection and union. Information Sciences **64**(219101), 1–219101 (2021)

[8] Mironov, I., Segev, G., Shahaf, I.: Strengthening the security of encrypted databases: Non-transitive joins. In: Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15, pp. 631–661 (2017). Springer

[9] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie–hellman assumptions. Journal of cryptology **30**, 242–288 (2017)

[10] Scott, M.: Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive (2002)

[11] Katz, J., Lindell, Y.: Introduction to Modern Cryptography: Principles

and Protocols. Chapman and hall/CRC, ??? (2007)

[12] Popa, R.A., Zeldovich, N.: Cryptographic treatment of cryptdb's adjustable join (2012)

[13] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88 (2006)

[14] De Caro, A., Iovino, V.: jpbc: Java pairing based cryptography. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 850–855 (2011). IEEE

[15] Su, J., Zhang, L., Mu, Y.: Ba-rmkabse: Blockchain-aided ranked multi-keyword attribute-based searchable encryption with hiding policy for smart health system. Future Generation Computer Systems **132**, 299–309 (2022)

[16] Wang, P., Chen, B., Xiang, T., Wang, Z.: Lattice-based public key searchable encryption with fine-grained access control for edge computing. Future Generation Computer Systems **127**, 373–383 (2022)

[17] Xie, M., Yang, X., Hong, H., Wei, G., Zhang, Z.: A novel verifiable chinese multi-keyword fuzzy rank searchable encryption scheme in cloud environments. Future Generation Computer Systems **153**, 287–300 (2024)

[18] Xu, Z., Tian, C., Zhang, G., Tian, W., Han, L.: Forward-secure multi-user and verifiable dynamic searchable encryption scheme within a zero-trust environment. Future Generation Computer Systems **166**, 107701 (2025)