



Flexible multi-client functional encryption for set intersection

Mojtaba Rafiee^{1,2}

Accepted: 21 February 2023 / Published online: 29 March 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

A multi-client functional encryption (MCFE) scheme [Goldwasser–Gordon–Goyal 2014] for set intersection is a cryptographic primitive that enables an evaluator to learn the intersection from all sets of a predetermined number of clients, without need to learn the plaintext set of each individual client. Using these schemes, it is impossible to compute the set intersections from arbitrary subsets of clients, and thus, this constraint limits the range of its applications. To provide such a possibility, we redefine the syntax and security notions of MCFE schemes, and introduce flexible multi-client functional encryption (FMCFE) schemes. We extend the aIND security of MCFE schemes to aIND security of FMCFE schemes in a straightforward way. For a universal set with polynomial size in security parameter, we propose an FMCFE construction for achieving aIND security. Our construction computes set intersection for n clients that each holds a set with m elements, in time $\mathcal{O}(nm)$. We also prove the security of our construction under DDH1 that it is a variant of the symmetric external Diffie–Hellman (SXDH) assumption.

Keywords Functional encryption · Secure computation · Set intersection · Multi-client · Flexible

1 Introduction

The dramatic growth of information, as well as increasing communications between different organizations to cover social activities in the digital world, has made the secure data sharing as a hot topic in the academic and industrial community.

✉ Mojtaba Rafiee
m.rafiie@mcs.ui.ac.ir

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), P.O. Box: 574619395, Tehran, Iran

In secure data sharing, the organizations tend to share their data in a controllable way such that users do not get more information. Any real-world computational function can be considered as an authorized control by a group of organizations that have shared their data. In the following, we present two examples of these functionalities in the practical environments:

- In order to track the prevalence of COVID-19 in European countries, an organization like ECDC needs the personal information of COVID-positive patients from the testing centers and hospitals where they were hospitalized, as well as the their acquaintances from the Civil Registry Office. In this example, our computational function is to extract the addresses of patients and their acquaintances from information records stored in the hospital and registry office, such that none of the other information fields are disclosed.
- A large-scale food advertising company needs to have information about consumer purchases in the chain stores across the country to deliver targeted and intelligent advertising. In this example, our computational function is to extract the mobile phone number and ten goods with the highest purchase request, such that other information such as volume of orders, value of orders and date of purchase is not revealed.

In general, the available solutions for secure data sharing can be divided into two categories: 1) *trivial solutions* and 2) *non-trivial solutions*. A brief description of each is given below.

Trivial solutions. At first glance, there are two naive solutions: 1) trivial full-trusted party and 2) trivial computing-trusted party, which are explained below.

- *Trivial full-trusted party.* In this setting, the organizations send their plain data to a third party that is reliable in terms of computing and storage. Then, the trusted party employs the considered functionality in the plaintext scenario to get the result and shares it with the authorized parties.
- *Trivial computing-trusted party.* In this setting, the trusted party has limited storage resources. Therefore, the organizations first encrypt their data and then outsource their encrypted data (instead of plain data) to the storage party (instead of the trusted party). Later, when needed, the trusted party in several steps, downloads the encrypted data from the storage party, decrypts them and employs the considered functionality in the plaintext scenario to get the intermediate results. Finally, the trusted party shares the final result with the authorized parties.

Non-trivial solutions. The academic and industrial researchers are trying to minimize the role of the trusted parties and ultimately eliminate these parties by providing distributed solutions. In this paper, we follow such an approach and assume that the *trusted parties* have limited storage and computation capabilities, and only perform the necessary coordination between organizations for a secure evaluation. In such a setting, the *organizations* encrypt their data using the parameters shared by the trusted party and send their encrypted data to the *storage party*. Then, each

Table 1 An asymptotic comparison of the different solutions for secure evaluation

Overheads	Parties	Trivial full-trusted	Trivial computing-trusted	Our expected solution
Storage	Organization	–	–	–
	Trust party	$\mathcal{O}(nm)$	–	–
	Storage	–	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$
	Evaluator	–	–	–
Communication	Organization	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$
	Trust party	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(1)$
	Storage	–	–	–
	Evaluator	–	–	$\mathcal{O}(m)$
Computation	Organization	–	–	–
	Trust party	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	–
	Storage	–	–	–
	Evaluator	–	–	$\mathcal{O}(nm)$

n : the total number of organizations, m : the maximum number of data values

authorized *evaluator* (such as participating organizations or even any other organization) that wants to perform an evaluation function first receives an evaluation key from the trusted party, then downloads the encrypted data from the storage party and finally computes the desired result.

Table 1 provides an asymptotic comparison of different solutions in terms of the storage, communication and computation overheads (for n organizations, each with m data values). In Table 1, the highlighted rows show the overheads imposed on the trusted party in each of the solutions described above.

Functional encryption (FE) schemes and their various types are the cryptographic tools that meet our requirements for the expected solution. These schemes provide a new paradigm for encryption that extend the traditional “all-or-nothing” requirement of the cryptosystems in a much more flexible way. In the following, we briefly introduce some of the most relevant ones.

- **The basic FE schemes.** A basic FE scheme [1, 2] is a cryptographic primitive that extends the decryption algorithm of the cryptosystems in a much more flexible way. In these scheme, the decryption algorithm requires a decryption key assigned to a 1-ary function f and a ciphertext ct computed for value x to output $f(x)$ (instead of the original value x from the ciphertext ct).
- **Multi-Input Functional Encryption (MIFE) schemes.** In the basic FE schemes, the function f is a 1-ary function. An extension on these schemes is to support the n -ary functions. The MIFE schemes [3] cover such a setting. In the MIFE schemes, the decryption algorithm needs a decryption key assigned to an n -ary function f and n ciphertexts ct_1, \dots, ct_n computed, respectively, for values x_1, \dots, x_n to output $f(x_1, \dots, x_n)$.

- **Multi-Client Functional Encryption (MCFE)schemes.** A strict subset of the MIFE schemes are called the MCFE schemes [3, 4]. In these schemes, the input components for the n -ary function f are labeled by a tag t and are independently provided by n distinct clients. Usually, the tag is the identifier of a time window in which clients generate their dataset. In these schemes, each set is labeled by a tag and the evaluation of function f can only be done on the sets with the same tag. Therefore, in such a setting, the decryption algorithm requires a decryption key assigned to an n -ary function f and n ciphertexts $ct_{t,1}, \dots, ct_{t,n}$ labeled for the same tag t (according to the values $x_{t,1}, \dots, x_{t,n}$, respectively) to output $f(x_{t,1}, \dots, x_{t,n})$.

Our desired scheme and functionality. In this paper, we specifically study the set intersection functionality as a function in a more flexible version of the MCFE schemes, called flexible multi-client functional encryption (FMCFE) schemes. These schemes support the set intersections from arbitrary subsets of clients (instead of a fixed set of the clients).

The following example intuitively summarizes and compares the schemes described above. Consider three different organizations and three distinct time windows. Each organization owns a set in each time window. For simplicity, we assumed that each set contains only one element. Also, we denote the elements as x_j^i , where i is organization identifier and j is tag identifier. For such a setting, Fig. 1 shows the computable results for different types of FE schemes along with their leakages.

As it can be seen, our FMCFE schemes have the most flexibility to obtain the desired results with the least leakage. More precisely, from the aspect of flexibility, the computable functions in FE, MIFE and MCFE have a fixed arity, while the computable functions in FMCFE schemes are flexible due to the variable arity. Consequently, schemes FE, MIFE and MCFE are a special case of FMCFE schemes. In terms of leakage, since the function generated in FE and MIFE can be applied to all ciphertexts with any desired combination, the size of the obtained results is uncontrollable and is far more than MCFE and FMCFE schemes.

In the following, we first review the researches related to MCFE schemes for set intersection. Then for the comprehensiveness of our discussion, we also introduce some other important cryptographic tools for our desired functionality.

1.1 The MCFE schemes for set intersection

Kamp et al. [5] proposed several multi-client functional encryption schemes for set intersection and its derivatives (such as set intersection cardinality, threshold set intersection and set intersection with data transfer). In their settings, they consider n clients and one evaluator. Each client labels his set with a tag, and then encrypts and holds it. The evaluator, by having a decryption key and receiving encrypted sets

x_i^j is data value of organization i for tag j	Computable results using FE schemes	Computable results using MCFE schemes
$x_1^1 \quad x_2^1 \quad x_3^1$ $x_1^2 \quad x_2^2 \quad x_3^2$ $x_1^3 \quad x_2^3 \quad x_3^3$	$f(x_1^1) \mid f(x_2^1) \mid f(x_3^1)$ $f(x_1^2) \mid f(x_1^1) \mid f(x_1^1)$ $f(x_1^1) \mid f(x_1^1) \mid f(x_1^1)$	$f(x_1^1, x_1^2, x_1^3) \mid f(x_2^1, x_2^2, x_2^3) \mid f(x_3^1, x_3^2, x_3^3)$
Computable results using MIFE schemes	Computable results using FMCFE schemes $(L=\{(1,2), (1,3), (2,3), (1,2,3)\})$	
$f(x_1^1, x_1^2, x_1^3) \mid f(x_1^1, x_1^2, x_3^3) \mid f(x_1^1, x_1^2, x_3^3)$ $f(x_1^1, x_2^2, x_1^3) \mid f(x_1^1, x_2^2, x_2^3) \mid f(x_1^1, x_2^2, x_3^3)$ $f(x_1^1, x_3^2, x_1^3) \mid f(x_1^1, x_3^2, x_2^3) \mid f(x_1^1, x_3^2, x_3^3)$ $f(x_2^1, x_1^2, x_1^3) \mid f(x_2^1, x_1^2, x_2^3) \mid f(x_2^1, x_1^2, x_3^3)$ $f(x_2^1, x_2^2, x_1^3) \mid f(x_2^1, x_2^2, x_2^3) \mid f(x_2^1, x_2^2, x_3^3)$ $f(x_2^1, x_3^2, x_1^3) \mid f(x_2^1, x_3^2, x_2^3) \mid f(x_2^1, x_3^2, x_3^3)$ $f(x_3^1, x_1^2, x_1^3) \mid f(x_3^1, x_1^2, x_2^3) \mid f(x_3^1, x_1^2, x_3^3)$ $f(x_3^1, x_2^2, x_1^3) \mid f(x_3^1, x_2^2, x_2^3) \mid f(x_3^1, x_2^2, x_3^3)$ $f(x_3^1, x_3^2, x_1^3) \mid f(x_3^1, x_3^2, x_2^3) \mid f(x_3^1, x_3^2, x_3^3)$	For $L=\{1,2\}$ $f(x_1^1, x_1^2) \mid f(x_1^1, x_2^2) \mid f(x_3^1, x_3^2)$ For $L=\{1,3\}$ $f(x_1^1, x_1^3) \mid f(x_2^1, x_2^3) \mid f(x_3^1, x_3^3)$ For $L=\{2,3\}$ $f(x_2^1, x_2^3) \mid f(x_2^2, x_2^3) \mid f(x_2^3, x_2^3)$ For $L=\{1,2,3\}$ $f(x_1^1, x_1^2, x_1^3) \mid f(x_2^1, x_2^2, x_2^3) \mid f(x_3^1, x_3^2, x_3^3)$	

Fig. 1 Computable results for different types of FE schemes

from the clients, can learn the result of considered operation on all sets without having to learn the plaintext set of each individual client. Their constructions support only a set intersection, and therefore, they do not need to define key generation algorithm to produce a decryption key. In the literature, these MCFE schemes are called single-key MCFE schemes.¹

Lee and Seo [6] proposed a multi-client functional encryption scheme for set intersection in multi-client setting. In their settings, they consider n clients and one evaluator (termed server). Each client labels his set with a tag, and then encrypts and outsources it to a server. Next, the server upon receiving a decryption key for a pair of sets can evaluate the set intersection for this pair. It should be noted that the construction proposed in [6], despite being provided for multi-client settings, and its decryption key allow the set intersection for each pair of clients, and therefore, it is different from the standard MCFE schemes that computes the set intersection for all sets of the clients. They also presented in [7] a decentralized version of MCFE

¹ In the single-key MCFE schemes, the decryption key is usually initialized along with other parameters used in the scheme in Setup algorithm.

Table 2 An asymptotic comparison of the existing constructions for a time step

Construction	Cryptographic tool	Assumption	Scenario model			Intersection time	Security level
			#client	arity	constraint		
Lee et al. [6]	Pairing-based	A variant of XDH [6]	$n = 2$	Fixed	–	$\mathcal{O}(m)$	aIND
Kamp et al. [5]	Group-based	DDH [9]	$n \geq 2$	Fixed	–	$\mathcal{O}(m^r)$	S-aIND
Libert et al. [8]	Group-based	LWE [10]	$n \geq 2$	Fixed	$ s_i = 1$	$\mathcal{O}(n)$	aIND
Chotard et al. [4]	Group-based	DDH [9]	$n \geq 2$	Fixed	$ s_i = 1$	$\mathcal{O}(n)$	aIND
Our construction	Pairing-based	DDH1 [11]	$n \geq 2$	Flexible	$ \mathcal{W} = \text{poly}(\lambda)$	$\mathcal{O}(nm)$	P-aIND

n : the total number of clients, m : the maximum size of each set

schemes for set intersection, in which the process of generating the decryption key is done with the participation of all clients and in a decentralized manner.

In addition to the mentioned constructions, Libert et al. in [8] and Chotard et al. in [4] have proposed MCFE schemes for linear functions that can be specifically applied to the set intersection. However, the security definitions proposed for these schemes restrict clients to having only single-element sets.

Table 2 summarizes the strengths and weaknesses of existing MCFE schemes for set intersection compared to our desired scheme, in terms of the cryptographic tool used, the underlying hardness assumption, the properties of the scenario model (the total number of the clients, intersection flexibility and considered constraints), the intersection time (for n sets each of size m) and security level.

1.2 Some similar schemes for set intersection

In this subsection, we present some schemes that are somewhat similar to our multi-client settings. In the following, each of these schemes is explained, and then, their similarities and differences with our scheme are presented.

Private Set Intersection (PSI) Protocols. The classic problem of private set intersection (PSI) protocols in the standard multi-party computation (MPC) [12] is somewhat similar to MCFE schemes for set intersection. The basic scenario model for these protocols includes several parties who each locally holds a private set. These parties interact with each other and perform set intersection. The goal is to compute the set intersection result in such a way that none of them is able to acquire any additional information besides what can be inferred from their own input and the computed result. A more advanced scenario model is “Delegated PSI” [13]. This scenario model considers a new party (termed server) compared to previous model. In this model, the parties outsource their sets to the server and take the advantages of its computational and storage superiority. Similar to the previous model, goal is to compute the result in such a way that none of the parties is able to obtain any additional information.

The similarities and differences. In a general view, both PSI protocols and MCFE schemes for set intersection have several parties, and the goal is to evaluate the set intersection of their sets. However, in the PSI scenario models, all parties learn the outcome of intersection, while in the MCFE schemes we require only one dedicated evaluator to only learn this outcome.

Multi-adjustable Join (M-Adjoin) Schemes. The M-Adjoin scheme, first proposed by Khazaei and Rafiee [14, 15], is a symmetric-key primitive that supports the secure join queries for a list of column labels on an encrypted database. The scenario model for this functionality consists of two main parties: a user and a server. The user outsources a database to the server, where the database contains a number of tables and each table includes several data records that are vertically partitioned into columns. When the user wishes to issue a join query on the database, he generates a join token and sends it to the server. A join query is formulated as a list of column labels. Finally, the server executes the requested join query on the encrypted database and returns the result to the user.

The similarities and differences. Here columns play the role of sets, and join queries play the role of set intersection. However, this scenario model is intended for single-user cloud scenarios (and no multi-user), all encrypted columns have the same tag, and its security notions [16] are different from the standard security notions of the MCFE schemes.

1.3 Challenges and contributions

In this paper, we are looking a solution for set intersection in multi-client setting that has the following features:

- In order to reduce the communication overheads, the evaluation of set intersection must be done non-interactively. Since FE schemes and its variants are designed for this purpose, and our solution is proposed in this framework, the challenge is addressed.
- In order to increase efficiency, the evaluation of set intersection must be done in polynomial-time complexity while no information is revealed except the intersection result. In the literature, the computation of set intersection in such a setting and without any constraints on the problem is known as an open problem and is investigated from a theoretical perspective [5, 14]. To overcome this problem, we consider a relaxed version in which size of universal set shared between clients is polynomial in security parameter.
- Another feature is the ability to publicly share sets of clients on a public bulletin board. With this feature, the clients can encrypt their sets in any time step and share it on the public board. Then, the evaluator can perform their evaluation at any time (no need for clients to be online). It should be noted that such a feature is not generally considered for the FE schemes and its variants.
- The last feature is the flexibility to compute the functions with arbitrary arities. The decryption key generated by MCFE schemes is applicable to all sets and not an arbitrary subset of them. To this end, we introduce flexible multi-client functional encryption (FMCFE) schemes as an extension of MCFE schemes, which provides such a feature.

In this paper, we have covered the above-mentioned features through:

- extending the syntax and security notions of the MCFE schemes, and introducing FMCFE schemes,
- extending the aIND security of MCFE schemes to a similar security notion for FMCFE schemes,
- for a universal set with polynomial size in security parameter, building a FMCFE scheme to achieve aIND security level,
- computing the set intersection for n clients, each containing m elements in time $\mathcal{O}(nm)$,
- proving security of our FMCFE construction in the random oracle model under the symmetric external Diffie–Hellman (SXDH) assumption,

- providing a concrete evaluation of our construction in the terms of time and space complexity.

1.4 Paper organization

The remaining of this paper is organized as follows. The problem statement and the preliminaries are explained in Sects. 2 and 3, respectively. We formalize the syntax and security of the FMCFE schemes for set intersection in Sect. 4. Our proposed FMCFE construction and its security are presented in Sect. 5 and Sect. 6, respectively. The performance analysis of our construction is provided in Sect. 7. Finally, Sect. 8 concludes the paper and points out future directions.

2 System model

Our scenario model considers four types of parties to securely evaluate the set intersection:

- **Client Parties (CP):** A group of parties who want to securely share their sets on a public bulletin board and allow to securely perform the set intersection on every arbitrary subset of their sets.
- **Evaluator Parties (EP):** The evaluators are the parties that allowed to compute the set intersection for a subset of sets shared on the public bulletin board. To this end, the evaluators download the desired sets and compute the outcome of the set intersection.
- **Storage Provider (SP):** The storage provider is a party that provides and manages the required storage for the public bulletin board.
- **Trusted Party (TP):** The trusted party is a party that determines and distributes the required parameters to securely share the sets owned by clients, and provides the required information to compute the set intersection for the evaluator.

Remark 1 In our model, each of the parties introduced above can play the role of evaluator. For example, responsibilities of the evaluator and storage provider can be performed by one party, simultaneously. This case is very similar to cloud scenario models where computation and storage facilities are outsourced to a cloud service provider (CSP).

Threat model. All of the parties introduced above, except the trusted party (TP), are assumed the honest-but-curious parties. A party 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 sets, requested set intersections and corresponding responses.

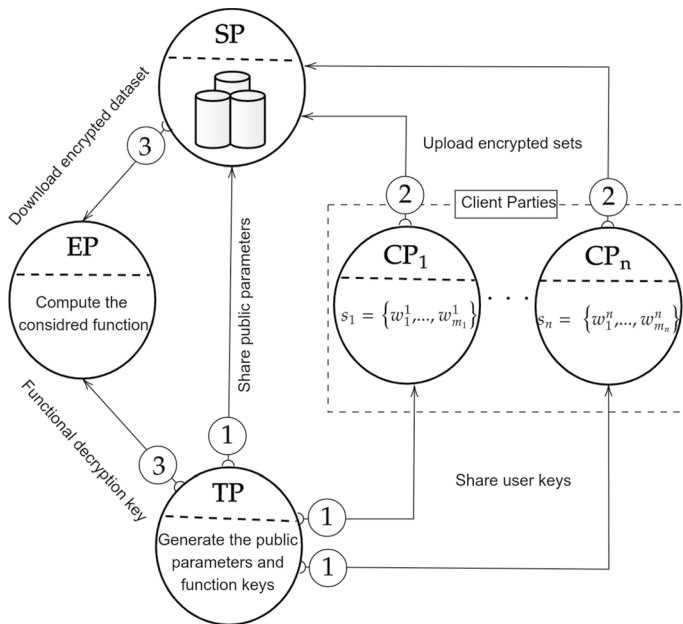


Fig. 2 An illustration of our system model

Desired functionality and flexibility. Before the detailed description of the problem statement, we describe the desired function and its expected property in the mentioned scenario model. The function that we are interested in this paper is set intersection, and the property that we expect for this function is the possibility of intersection on any arbitrary subset of dataset (instead of a fixed size). We call such functions flexible function.

Suppose we have 10 sets s_1 to s_{10} . For a function f with constant arity 10, it is only possible to calculate f on the entire sets s_1 to s_{10} , and it is not possible on an arbitrary subset of the dataset. It is clear that flexible functions are more general form compared to fixed-arity functions, and therefore cover a wider range of applications.

Problem statement. Suppose that \mathcal{W} is the universal set of all set elements available to the clients, and $s_i \subseteq \mathcal{W}$ is a set belong to i th client that holds it locally. Our problem is to design a cryptographic primitive, according to the scenario model and the threat model described above, that enables the clients to securely share their sets, and to evaluate the outcome of the set intersection for every arbitrary subset of them. Figure 2 shows an illustration of our system model.

3 Preliminaries

In this section, we introduce some notations and basic cryptographic primitives that are used throughout the paper.

3.1 Notation

Throughout the paper, we consider the symbol λ to denote the security parameter. We use $[n]$ to denote the set $\{0, 1, \dots, n\}$, where n is a positive integer. Let A is a (possibly) probabilistic algorithm, $y \leftarrow A(x)$ shows that y is the output of the algorithm A on x . We use the abbreviation PPT for probabilistic polynomial time. Suppose that S is a finite set, $x \leftarrow S$ means that the element x selected as uniform from the set S . We say that a function is negligible and denotes it by negl , if it is smaller than the inverse of any polynomial in the security parameter λ for sufficiently large values of λ . As a convention, we denote the output of a defined experiment by the experiment name itself. We use the symbol $|$ to denote the concatenation of bit strings (e.g., $010 | 101 = 010101$).

3.2 Basic primitives

In this subsection, we review some basic cryptographic primitives that are used in our construction. The readers familiar with these primitives can safely skip this subsection.

Pseudorandom Function (PRF). Let X, Y be two sets. A polynomial-time computable function $F : \{0, 1\}^\lambda \times X \rightarrow Y$ is a pseudorandom function, if for every PPT adversary \mathcal{A} , we have:

$$|\Pr[k \leftarrow \{0, 1\}^\lambda : \mathcal{A}^{F_k(\cdot)}(1^\lambda) = 1] - \Pr[f \leftarrow \text{RF} : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]| \leq \text{negl}(\lambda),$$

where RF is the set of all the functions from X to Y .

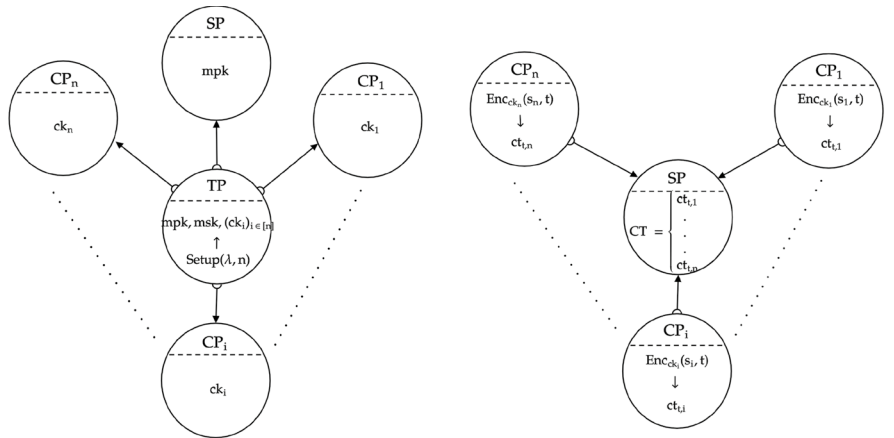
Collision resistance hash function. Function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ is a collision resistance hash function, if it holds the following properties:

- The function $H(x)$ is easy to compute for any x ,
- It is hard to find $x \neq x'$ such that $H(x) = H(x')$.

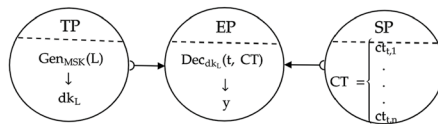
Bilinear map: Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order q , and g_1, g_2 are generators for $\mathbb{G}_1, \mathbb{G}_2$, respectively. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that satisfies the following properties:

1. Bilinearity: $\forall x, y \in \mathbb{Z}_q : e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$,
2. Non-degeneracy: $e(g_1, g_2) \neq 1$,
3. Computability: e can be computed efficiently.

We assume that we have an efficient bilinear map generator such as \mathcal{G} that on the security parameter λ as input, outputs tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$.



(a) Generate the parameters by the trusted party (b) Encrypt sets and share them by each client



(c) Download encrypted sets and compute the set intersection by the evaluator.

Fig. 3 Generate the parameters by the trusted party

4 FMCFE-SI scheme

In this section, we define the syntax and security notions of FMCFE schemes for set intersection. For convenience, hereafter we call these schemes FMCFE-SI. It should be noted that we only define the FMCFE-SI schemes in private-key setting, and leave the study of the FMCFE-SI schemes in the public-key setting for future works. In the following, each of syntax and security definitions are first explained informally and then the formal definitions are provided.

4.1 FMCFE-SI syntax

A FMCFE-SI is a symmetric primitive that enables a group of predetermined clients to securely share their sets, and also enables an evaluator to learn the set intersection of every arbitrary subset of these sets, without having to learn the plaintext set of each individual client.

Figure 3, for n clients, shows how to use FMCFE-SI schemes. At first, the trusted party generates a set of public parameters pp , a master secret key msk and a list of client keys $(ck_i)_{i=1}^n$ using a key generation algorithm denoted by Setup . Then, for

every $i \in [n]$, the trusted party sends ck_i to the client with identifier i and shares pp on the public bulletin board (Fig. 3a). Next, each client encrypts its set using an encryption algorithm denoted by Enc and sends it to the storage provider (Fig. 3b). Later, when the trusted party wants to send a functional decryption key to the evaluator, on demand, he selects a time window t and a list of clients $L \subseteq [n]$, and calls the key generation algorithm denoted by Keygen . Finally, the evaluator downloads the requested sets and computes the outcome of the set intersection using a decryption algorithm denoted by Dec (Fig. 3c).

In the following, we provide a formal definition of the FMCFE-SI schemes.

Definition 1 (FMCFE-SI syntax) A FMCFE-SI scheme is a collection of four polynomial-time algorithms $\Pi = (\text{Setup}, \text{Enc}, \text{Keygen}, \text{Dec})$ such that:

- $(pp, msk, (ck_i)_{i=1}^n) \leftarrow \text{Setup}(\lambda, n)$: takes as input a security parameter λ and a pre-determined number of the clients n , and returns a set of public parameters pp , a master secret key msk and a list of client keys $(ck_i)_{i=1}^n$.
- $ct_{t,i} \leftarrow \text{Enc}(ck_i, s_i, t)$: takes as input the client key ck_i , a set s_i and a tag t , and returns a ciphertext $ct_{t,i}$.
- $dk_L \leftarrow \text{Keygen}(msk, L)$: takes as input the master secret key msk and a list of client identifiers L , and outputs a functional decryption key dk_L .
- $y \leftarrow \text{Dec}(dk_L, t, CT)$: takes as input a decryption key dk_L , a tag t and a $|L|$ -list ciphertext CT . It returns as output $y = \bigcap_{i \in L} s_i$ if CT is a valid encryption of sets $(s_i)_{i \in [L]}$ for tag t , or $y = \perp$ otherwise.

Correctness. The FMCFE-SI scheme is said to be correct, if for any integer $n \geq 2$, any list $L \subseteq [n]$ with size $|L| \geq 2$, any tag t and any list of sets $(s_i)_{i \in L}$, it holds that:

$$\Pr \left[\begin{array}{l} (pp, msk, (ck_i)_{i=1}^n) \leftarrow \text{Setup}(\lambda, n); \\ \forall i \in [L] \quad ct_{t,i} \leftarrow \text{Enc}(ck_i, s_i, t); \\ dk_L \leftarrow \text{Keygen}(msk, L) : \\ \text{Dec}(dk_L, t, CT) = \bigcap_{i \in L} s_i \end{array} \right] = 1.$$

4.2 FMCFE-SI security

In this subsection, we formalize the security notions of FMCFE-SI schemes based on the indistinguishability games. To this end, we have taken ideas from the security definitions proposed in [17] to handle the flexibility of the function arity, and also we have taken ideas from the security definitions presented in [4] to handle the time step in our FMCFE-SI schemes.

In our security game, we need that the encrypted elements of the clients do not reveal any information about their plaintext elements. We also need that by having the evaluation key for a subset of sets, only the intersection of this subset can be computed and no information beyond it can be revealed. Additionally, our game consider three capabilities for the adversaries: 1) the adaptive queries to get the

evaluation key for each desired subset, 2) the adaptive queries to encrypt any element from each client and 3) the adaptive queries to corrupt from each client. In this game, we also consider the constraints that cause the adversary to easily not be able to win the game. In the following, we provide a formal definition of our game.

The adaptive indistinguishability-based (aIND) security game $\text{Exp}_{\mathcal{A}, \Pi}^{\text{aIND}}(\lambda)$:

1. **Initialization phase:** The challenger runs $(pp, msk, (ck_i)_{i=1}^n) \leftarrow \text{Setup}(\lambda, n)$ and selects a random bit $b \leftarrow \{0, 1\}$. Also, the challenger considers a set \mathcal{HS} of honest clients (initialized to $\mathcal{HS} = [n]$) and a set \mathcal{CS} of corrupted clients (initialized to $\mathcal{CS} = \emptyset$).
2. **Pre-challenge query phase:** The adversary \mathcal{A} may adaptively issue $\text{Enc}(\cdot, \cdot, \cdot)$, $\text{Keygen}(\cdot)$ and $\text{Corrupt}(\cdot)$ queries, which are defined as follows:
 - (a) $\text{Enc}(i, s_i, t)$: The challenger computes and returns to the adversary \mathcal{A} a ciphertext $ct_{t,i} \leftarrow \text{Enc}(ck_i, s_i, t)$. For any given pair (i, t) , only one query is allowed and later queries involving the same pair (i, t) are ignored.
 - (b) $\text{Keygen}(L)$: The challenger runs the key generation algorithm $dk_L \leftarrow \text{Keygen}(msk, L)$ and returns to the adversary the decryption key dk_L .
 - (c) $\text{Corrupt}(i)$: The challenger adds i to \mathcal{CS} (i.e., $\mathcal{CS} = \mathcal{CS} \cup \{i\}$), removes i from the set \mathcal{HS} (i.e., $\mathcal{HS} = \mathcal{HS} \setminus \{i\}$) and returns to the adversary \mathcal{A} the client key ck_i .
3. **Challenge query phase:** The adversary \mathcal{A} adaptively issues challenge queries of the form $\text{Enc}(i, s_{i,0}^*, s_{i,1}^*, t^*)$ and as a response obtains a ciphertext $ct_{t^*,i} \leftarrow \text{Enc}(ck_i, s_{i,b}^*, t^*)$. It should be noted that in this phase, only one tag t^* can be queried and also similar to the pre-challenge phase, query for the same pair (i, t^*) will later be ignored.
4. **Post-challenge query phase:** Identical to the pre-challenge phase.
5. **Finalize phase:** The adversary \mathcal{A} outputs a value $\hat{b} \in \{0, 1\}$ which is defined as the output of the experiment.

Valid adversary. We say that the adversary \mathcal{A} is a valid adversary for the game $\text{Exp}_{\mathcal{A}, \Pi}^{\text{aIND}}(\lambda)$, and if for every security parameter λ , in all transcripts of the game $\text{Exp}_{\mathcal{A}, \Pi}^{\text{aIND}}(\lambda)$, it holds that for every queried list $L = (i_1, \dots, i_l)$, there does not exist

$$\bigcap_{i \in L} s_{i,0}^* \neq \bigcap_{i \in L} s_{i,1}^*,$$

where for every $i \in L$, we have

- $i \in \mathcal{CS}$; therefore, there is no constraint on $s_{i,0}^*$ and $s_{i,1}^*$.
- There is a challenge query of the form $\text{Enc}(i, s', s'', t^*)$ such that $s_{i,0}^* = s'$ and $s_{i,1}^* = s''$.

Definition 2 (FMCFE-SI security) A FMCFE-SI scheme such as $\Pi = (\text{Setup}, \text{Enc}, \text{Keygen}, \text{Dec})$ is aIND-secure if for every PPT valid adversary \mathcal{A} , there exists a negligible function negl such that

$$\text{Adv}_{\mathcal{A}, \Pi}^{\text{aIND}} = |\Pr[\text{Exp}_{\mathcal{A}, \Pi}^{\text{aIND}}(\lambda) = 1] - \frac{1}{2}| \leq \text{negl}(\lambda).$$

Weaker security notions. Similar to [4], we can consider two weaker security notions for the FMCFE-SI schemes:

- **Passive security** (P-aIND): The P-aIND security is defined similar to the aIND security except that no corruption query is issued. We show the game of this security notion with $\text{Exp}_{\mathcal{A}, \Pi}^{\text{P-aIND}}$.
- **Static security** (S-aIND): The S-aIND security is defined similar to the aIND security except that all corruption queries are sent before the initialization phase. We denote the game of this security notion by $\text{Exp}_{\mathcal{A}, \Pi}^{\text{S-aIND}}$.

Generally, the various features and functionalities that can be provided by a new cryptographic scheme are reasons that make it difficult to propose a construction with strong security notion, and lead us to weaker security notions. In this regard, past research in the literature confirms this and shows that the constructions with weaker security notions are first introduced, and then, steps are gradually taken to provide constructions with stronger security.

In this paper, based on the reasons mentioned above, we follow a similar approach and present a FMCFE-SI with P-aIND security level.

5 Our FMCFE-SI construction

In this section, we propose a FMCFE-SI construction that supports a universal set of polynomial size in the security parameter λ , and in Sect. 6, we prove that it is P-aIND-secure. Our construction use a bilinear group generator \mathcal{G} that takes as input the security parameter λ and returns a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$, where q is a λ -bit prime number, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order q , $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear map, and g_1, g_2 are generators of \mathbb{G}_1 and \mathbb{G}_2 , respectively. In addition, our construction uses a cryptographic hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$. The algorithms of our FMCFE-SI construction are defined as follows:

- $(\text{pp}, \text{msk}, (\text{ck}_i)_{i=1}^n) \leftarrow \text{Setup}(\lambda, n)$: On input of the security parameter λ and a number of clients n , it acts as follows:
 1. It runs bilinear map generator $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e) \leftarrow \mathcal{G}(\lambda)$.
 2. For every $i \in [n]$, it samples a client key ck_i .

3. It defines the master secret key $msk = (ck_i)_{i=1}^n$ and the public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$.
 4. It returns as output $(pp, msk, (ck_i)_{i=1}^n)$.
- $ct_{t,i} \leftarrow \text{Enc}(ck_i, s_i, t)$: Given the client key ck_i , a set s_i and a tag t , it works as follows:
 1. For every $w \in \mathcal{W}$, it computes $ct_{t,i}[w]$ as follows:
 - if $w \in s_i$, it computes $ct_{t,i}[w] = (H(w \parallel t))^{ck_i}$.
 - if $w \notin s_i$, it selects a random value $r \leftarrow \mathbb{Z}_q^*$ and then defines $ct_{t,i}[w] = g_1^r \in \mathbb{G}_1$.
 - $dk_L \leftarrow \text{Keygen}(msk, L)$: On input of the master secret key msk and a list of clients L , it computes dk_L as follows:
 1. It parses msk as $(ck_i)_{i=1}^n$ and L as (i_1, \dots, i_l) .
 2. For every $i \in L$, it samples $z_i \leftarrow \mathbb{Z}_q^*$ such that $\sum_{i \in L} z_i = 0$.
 3. For every $i \in L$, it computes $at_i \leftarrow g_2^{z_i ck_i^{-1}}$ and finally defines $dk_L = (at_i)_{i \in L}$.
 - $y \leftarrow \text{Dec}(dk_L, t, CT)$: Given the decryption key dk_L , a tag t and a $|L|$ -list ciphertext CT , it computes y as follows:
 1. It parses dk_L as $(at_i)_{i \in L}$ and CT as $(ct_{t,i})_{i \in L}$.
 2. If $\prod_{i \in L} e(ct_{t,i}[w], at_i) = 1$, then it appends w to y , for every $w \in \mathcal{W}$.

Remark 2 As it can be seen in Algorithm Dec, this algorithm parameterized based on dk_L and t , and it can only compute the result of set intersection related to the clients involved in dk_L . Therefore, the decryption of the our scheme is different from the decryption of common symmetric/asymmetric encryption schemes.

Correctness. For any integer $n \geq 2$, any list $L \subseteq [n]$ with size $|L| \geq 2$, any tag t , any list of sets $(s_i)_{i \in L}$ and any $w \in \mathcal{W}$, it holds that:

$$\prod_{i \in L} e(ct_{t,i}[w], at_i) = \prod_{i \in L} e(g_1^{\hat{w}_{t,i} ck_i}, g_2^{z_i ck_i^{-1}}) = \prod_{i \in L} e(g_1, g_2)^{\hat{w}_{t,i} z_i} \quad (1)$$

where $(pp, msk, (ck_i)_{i=1}^n)$ is the output of $\text{Setup}(\lambda, n)$, the ciphertext $ct_{t,i}$ is the output of $\text{Enc}(ck_i, s_i, t)$ for any $i \in L$, the decryption key $dk_L = (at_i)_{i \in L}$ is the output of $\text{Keygen}(msk, L)$, and $(z_i)_{i \in L}$ are random values from \mathbb{Z}_q^* (s.t. $\sum_{i \in L} z_i = 0$).

Therefore, if $w \in \bigcap_{i \in L} s_i$, using Eq. 1 we have:

$$\prod_{i \in L} e(g_1, g_2)^{\hat{w}_i z_i} = e(g_1, g_2)^{\sum_{i \in L} \hat{w}_i z_i} = e(g_1, g_2)^{\hat{w}_t \sum_{i \in L} z_i}, \quad (2)$$

and since $\sum_{i \in L} z_i = 0$, we have $e(g_1, g_2)^{\hat{w}_t \sum_{i \in L} z_i} = 1$, and finally, w belongs to the result set.

Moreover, if there is $i, j \in L$ such that $w \in s_i$ and $w \notin s_j$, then with a overwhelming probability we have $H(w, t)^{ck_i} \neq g_1^r$, where r is a random value from \mathbb{Z}_q^* . It is easy to show that the probability that $e(g_1, g_2)^{\sum_{i \in L} \hat{w}_i z_i} = 1$ (i.e., $w \in \text{Dec}(dk_L, t, CT)$) is at most $\frac{1}{q} + \text{negl}(\lambda)$, where $\text{negl}(\lambda)$ is some negligible function and q is the range size of hash function H . The claim then follows, since q is exponential in the security parameter λ . \square

In the following, we show how the above construction cover the challenges raised in Subsection 1.3.

Minimal client interaction. As it can be seen in Algorithms Setup and Enc, each client with his secret key (which is determined by Algorithm Setup) can encrypt and share his sets independently and without interacting with other clients. As a result, due to the non-interaction of clients with each other, the overheads imposed by sharing sets are reduced.

Evaluation with offline clients. After sharing the encrypted sets by clients, as it can be seen in Algorithm Dec, the clients do not need to be online to compute the set intersection.

Flexible functionality. As it can be seen in Algorithm Keygen, due to the use of secret sharing scheme ideas, it is possible to generate decryption keys to be applied to any arbitrary subset of the dataset. Therefore, with such decryption keys, flexible intersection is guaranteed.

Efficient construction. Assuming that the size of the universal set is polynomial in terms of the security parameter, as it can be seen in Algorithm Dec, our evaluation method for m sets each of size n is of the order $\mathcal{O}(mn)$. More precisely, during encryption, each set is appropriately expanded to the size of the universal set, and then, during evaluation, it should be checked for each element of the universal set whether this element exists in the queried sets or not. It should be noted that in the literature, the problem of computing the set intersection in polynomial time, without restrictions on the size of the universal set and without disclosing any information except the intersection result, is known as an open problem [5, 14].

6 Security analysis

In this section, we first review a well-known computational hardness assumption used to prove the security of our FMCSE-SI construction. Then, we prove that our construction is P-aIND-secure.

The symmetric external Diffie–Hellman (SXDH) assumption, formalized in [11, 18–20], is a computational hardness assumption that underlies the security of several pairing-based cryptosystems such as [21, 22]. Our FMCFE-SI construction is proved secure under DDH1 that it is a variant of the SXDH assumption.

Assumption 1 (DDH1) Decisional Diffie–Hellman assumption in \mathbb{G}_1 (DDH1) for the bilinear map generator \mathcal{G} states that it is hard to distinguish g_1^{am} from a random group element g_1^r , when given g_1, g_2 , and random group elements g_1^a and g_1^m .

The dual of Assumption 1 is decisional Diffie–Hellman assumption in \mathbb{G}_2 (DDH2), which is identical to the above assumption with the roles of \mathbb{G}_1 and \mathbb{G}_2 reversed. We say that symmetric external Diffie–Hellman (SXDH) assumption holds for the bilinear map generator \mathcal{G} , if DDH problems are intractable in both \mathbb{G}_1 and \mathbb{G}_2 .

Theorem 1 If H be a collision-resistant hash function modeled as random oracle, and the DDH1 assumption holds relative to \mathcal{G} , then our FMCFE-SI construction of Sect. 5 is P-aIND-secure.

Proof To prove the security of the proposed construction, we use the experiment defined in subsection 4.2, the computational hardness assumption DDH1 and some standard proof techniques in the literature. In particular, using technique “Proof By Reduction,” we show that if there is a successful attack on our construction, then the hard problem DDH1 will be solved and no longer hard. In other words, we tie the solution of the hard problem to maintaining the security of our construction. Therefore, since problem DDH1 is difficult to solve, the security of the proposed construction is also preserved. The details of the proof are given below.

We need to show that advantage of every PPT valid adversary \mathcal{A} in the game $\text{Exp}_{\mathcal{A}, \Pi}^{\text{P-aIND}}(\lambda)$ is negligible.

In the following, we describe an algorithm that is able to break the DDH1 problem, if the valid adversary \mathcal{A} has a non-negligible advantage in winning the game $\text{Exp}_{\mathcal{A}, \Pi}^{\text{P-aIND}}(\lambda)$.

The challenger simulates the random function f and key values $(ck_i)_{i \in L}$ as follows (without knowing values m and a explicitly):

$$H(t, w) = \begin{cases} (g_1^m)^{h_{t,w}} & \text{if } t = t^* \\ g_1^{h_{t,w}} & \text{otherwise} \end{cases}, \quad (3)$$

$$ck_i = a \cdot a'_i, \quad (4)$$

where $h_{t,w}$ and $(a'_i)_{i \in [n]}$ are the random values from \mathbb{Z}_q^* . It should be noted that based on the desired security, in which no corruption is considered for the clients, no keys are provided to the clients, and the keys only appear in the encryption and token algorithms in the form of powers of g_1 . Therefore, as we will see below, the challenger can easily simulate the computations required on these keys. In the following, we describe the challenger in details:

- **Initialization phase.** Given a bilinear group description $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$ and a tuple (g_1^a, g_1^m, g_1^s) from the DDH1 problem, where $s = am$ or $s = z$, the challenger acts as follows:
 1. The challenger defines a set of the public parameters as $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$.
 2. For every $i \in [n]$, the challenger samples a'_i from \mathbb{Z}_q^* and defines $ck_i = a \cdot a'_i$.
 3. Finally, the challenger samples $b \leftarrow \{0, 1\}$, defines the master secret key $msk = (ck_i)_{i \in [n]}$ and initializes $\mathcal{HS} = [n]$ and $\mathcal{CS} = \emptyset$.
- **Pre-challenge query phase.** In the following, we determine how the challenger handles the adversary's queries in the pre-challenge:
 - **Enc($\mathbf{i}, \mathbf{s}_i, \mathbf{t}$)** : The challenger computes $ct_{t,i}$ as follows: [1.] For every $w \in \mathcal{W}$ acts as follows: – For pair (t, w) , the challenger selects $f_{t,w} \leftarrow \mathbb{Z}_q^*$, unless it has already been sampled. – If $w \in s_i$, then the challenger computes $ct_{t,i}[w] = (g_1^a)^{a'_i f_{t,w}}$. – If $w \notin s_i$, then challenger selects a random value r from \mathbb{Z}_q^* and computes $ct_{t,i}[w] = g_1^r$. [2.]
 - Finally, the challenger returns $ct_{t,i}$ to the valid adversary \mathcal{A} .
 - **Keygen(\mathbf{L})** : The challenger computes dk_L as follows: [1.] The challenger selects a list of random values $(z_i)_{i \in L}$ from \mathbb{Z}_q^* such that $\sum_{i \in L} z_i = 0$. [2.] For every $i \in L$, the challenger computes $at_i = (g_2^{a_i^{-1}})^{r_i}$, where $r_i = a^{-1} z_i$. Note that we do not directly have the value of a^{-1} to compute the value of the tokens at_i , but we can easily assume that this value is absorbed in the random value of r_i . [3.] Finally, the challenger returns $dk_L = (at_i)_{i \in L}$.
 - **Challenge query phase.** For every $i \in [n]$, upon receiving the challenge queries $\text{Enc}(i, s_{i,0}^*, s_{i,1}^*, t^*)$, the challenger computes $ct_{t^*,i}$ as follows: [1.] For every $w \in \mathcal{W}$, the challenger acts as follows: – For tuple (t^*, w) , the challenger selects $f_{t^*,w} \leftarrow \mathbb{Z}_q^*$, unless it has already been sampled. – If $w \in s_{i,b}^*$, then the challenger computes $ct_{t^*,i}[w] = (g_1^s)^{a'_i f_{t^*,w}}$. – If $w \notin s_{i,b}^*$, then challenger selects a random value r from \mathbb{Z}_q^* and computes $ct_{t^*,i}[w] = g_1^r$. [2.] Finally, the challenger returns $ct_{t^*,i}$ to the adversary \mathcal{A} . Based on the details described above, the challenger of Game $\text{Exp}_{\mathcal{A}, \Pi}^{\text{P-aIND}}$ successfully performs its simulation because:

The challenger correctly generates the input and output distributions of all algorithms,

The challenger uses all parts of the computational hardness assumption correctly: – It uses g_1^m to produce the outputs of hash function H , – It uses g_1^a to produce the encrypted sets and decryption keys, – It uses g_1^s to produce the encrypted challenge set.

Note that if $s = am$, the ciphertext is distributed properly according to the scheme, and if $s = z$, then the challenger returns a ciphertext of a randomly distributed set element.

Table 3 A concrete evaluation of our construction

Algorithm	#exp	#hash	#mapping	Time (milliseconds)	Size (byte)
Setup	–	–	–	1124	979
Enc (for every set element)	1	1	–	0.4	40
Keygen (for every selected set)	1	–	–	3	120
Dec (adjust and compare for any two elements)	–	–	2	$3 + 2 \times 10^{-4}$	240

Table 4 Effect of limiting the size of the universal set

#client	$n = 2$	$n = 4$	$n = 6$	$n = 8$	$n = 10$
With constraint ($ \mathcal{W} = \text{poly}(\lambda)$)	6 s	12 s	19 s	25 s	30 s
Without constraint	6.2 s	~ 72 h	~ 6000 years	~ 6 billion years	~ 6000 billion years

- **Post-challenge query phase.** The challenger replies the adversary's queries similar to the pre-challenge phase.

7 Performance analysis

In this section, we provide a concrete evaluation of effective components in our construction in terms of the latency and the output size.

Since that the existing constructions have different security levels and different flexibility compared to our construction, and also because our construction is based on bilinear maps which has a moderate overhead compared to other cryptographic primitives, our goals of performance analysis are summarized in:

- Finding a good view of the execution time of the effective components in our construction,
- Highlighting the execution time with/without constraint on the size of universal set.

We implemented our construction in Java and used the jPBC library [23] for implementation of a Type-D curve (parameter d159) for the pairing setting. The evaluations are done on an Ubuntu 17.04 desktop PC with an Intel Processor 2.9 GHz. Table 3 shows a concrete evaluation of our construction in terms of the latency and the output size related to: the setup algorithm, the encryption algorithm (for every set element), the key generation algorithm (for each set involved in the set intersection) and the decryption algorithm (adjust and compare for any two elements in \mathbb{G}_T).

Table 4, based on the execution times given in Table 3, shows the effect of the constraint on the size of the universal set for different ranges of clients that each holds a set of size 1000. As an example in this table, a setting with 6 clients that each holds a set of size 1000, the existing constructions with polynomial size constraint need to approximately spend 19 seconds while without this constraint need to approximately spend about 6000 years. As a result, this relaxation is necessary to obtain the desired security as well as applying it in practical applications.

8 Conclusions and future works

In this paper, we first introduced a more flexible version of the MCFE schemes for set intersection, called the FMCFE-SI scheme, where an evaluator can learn the outcome of the set intersection for every arbitrary subset of a predetermined number of clients (instead of all clients). In addition, for a universal set with polynomial size in security parameter, we proposed an efficient FMCFE-SI construction for achieving P-aIND security level. Our construction computes set intersection for n clients that each holds a set with m elements, in time $\mathcal{O}(nm)$.

Future contributions can be made in aspects such as proposing the FMCFE-SI schemes that satisfy aIND security in the standard model, developing the FMCFE-SI schemes for a decentralized setting in which the trusted party is eliminated and the clients work together to generate the decryption keys, and providing the FMCFE-SI schemes with polynomial-time complexity without constraint on the size of universal set.

Acknowledgements This research was in part supported by a grant from IPM (No. 1401940052).

Author Contributions This declaration is not applicable. (This is a single-authored article.)

Funding This declaration is not applicable.

Availability of data and materials This declaration is not applicable.

Declarations

Conflict of interest We declare that have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval This declaration is not applicable.

References

1. Boneh D, Sahai A, Waters B (2011) Functional encryption: definitions and challenges. Springer, London, pp 253–273
2. Sahai A, Waters B (2005) Fuzzy identity-based encryption. Springer, London, pp 457–473
3. Goldwasser S, Gordon SD, Goyal V, Jain A, Katz J, Liu F-H, Sahai A, Shi E, Zhou H-S (2014) Multi-input functional encryption. Springer, London, pp 578–602

4. Chotard J, Sans ED, Gay R, Phan DH, Pointcheval D (2018) Decentralized multi-client functional encryption for inner product. In: International Conference on the Theory and Application of Cryptology and Information Security. (Springer: London, pp 703–732)
5. van de Kamp T, Stritzl D, Jonker W, Peter A (2019) Two-client and multi-client functional encryption for set intersection. Australasian Conference on Information Security and Privacy. Springer, Germany, pp 97–115
6. Lee K, Seo M (2022) Functional encryption for set intersection in the multi-client setting. *Des Codes Cryptog* 90(1):17–47
7. Lee K (2022) Decentralized multi-client functional encryption for set intersection with improved efficiency. *Des Codes Cryptog* 102:1–41
8. SD, L.B.T.R.G., Moriai, S (2019) Multi-client functional encryption for linear functions in the standard model from lwe. In: *Advances in Cryptology–ASIACRYPT*. 2019
9. Boneh D (1998) The decision diffie-hellman problem. International algorithmic number theory symposium. Springer, Berlin, pp 48–63
10. Regev O (2009) On lattices, learning with errors, random linear codes, and cryptography. *J ACM (JACM)* 56(6):1–40
11. Scott M (2002) Authenticated id-based key exchange and remote log-in with simple token and PIN number. *IACR Cryptol ePrint Archive* 2002:164
12. Freedman MJ, Nissim K, Pinkas B (2004) Efficient private matching and set intersection. Springer, Heidelberg, pp 1–19
13. Dong C, Chen L, Camenisch J, Russello G (2013) Fair private set intersection with a semi-trusted arbiter. IFIP Annual Conference on Data and Applications Ssecurity and Privacy. Springer, Germany, pp 128–144
14. Khazaei S, Rafiee M (2020) Towards more secure constructions of adjustable join schemes. *IEEE Trans Depend Sec Comput* 19(2):1079–1089
15. Rafiee M, Khazaei S (2021) Private set operations over encrypted cloud dataset and applications. *Comput J* 64(8):1145–1162
16. Rafiee M, Khazaei S (2021) Security of multi-adjustable join schemes: Separations and implications. *IEEE Trans Depend Secure Comput* 19(4):2535–2545
17. Agrawal S, Clear M, Frieder O, Garg S, O'Neill A, Thaler J (2020) Ad hoc multi-input functional encryption. In: 11th Innovations in Theoretical Computer Science Conference (ITCS 2020) . Schloss Dagstuhl-Leibniz-Zentrum für Informatik
18. Ballard L, Green M, de Medeiros B, Monrose F (2005) Correlation-resistant storage via keyword-searchable encryption. *IACR Cryptol ePrint Arch* 2005:417
19. Galbraith SD, Rotger V (2004) Easy decision diffie-hellman groups. *LMS J Comput Math* 7:201–218
20. Boneh D, Boyen X, Shacham H (2004) Short group signatures. Springer, London, pp 41–55
21. Ateniese G, Camenisch J, de Medeiros B (2005) Untraceable RFID tags via insubvertible encryption. pp 92–101
22. Camenisch J, Hohenberger S, Lysyanskaya A (2005) Compact e-cash, pp 302–321
23. De Caro A, Iovino V (2011) jpbcc: Java pairing based cryptography. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp 850–855

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.