A Fast and Flexible Multi-Client Functional Encryption for Set Intersection

Mojtaba Rafiee

School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Abstract. A Multi-Client Functional Encryption (MCFE) scheme for set intersection is a cryptographic primitive that enables an evaluator to learn the intersection from all sets of a pre-determined number of clients, without having to learn the plaintext set of each individual client. In this paper, we propose a flexible version of the MCFE schemes for the set intersection, called Flexible Multi-Client Functional Encryption for Set Intersection (FMCFE-SI). In these schemes, the evaluator can learn the intersection from any flexible choice of sets (instead of all sets). In this regard, we redefine syntax and security notions of the MCFE schemes for the FMCFE schemes. In the literature, solving multi-client set intersection problem in polynomial time, such that only the intersection result is revealed (without additional information), is an open problem. In this paper, we propose a relaxed solution using FMCFE-SI schemes to solve secure set intersection in polynomial time. We analyze that for practical use of secure multi-client set intersection, this relaxation is necessary. We also show that our scheme has the adaptive indistinguishability-based security under passive corruption. Our proof relies on the Symmetric eXternal Diffie-Hellman (SXDH) assumption in the standard model.

Keywords: Functional Encryption · Set Intersection · Secure Computation, 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.

In secure data sharing, the organizations tend to share their data in a controllable way. In other words, the organizations want the users to get only the information that is allowed, not 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 describe three 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 relatives from the Civil Registry Office.
- Corona virus vaccine production companies need to collect the genomic information of the patients from different countries in order to know the effectiveness of their vaccines.
- 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.

There are two naive solutions for secure data sharing: 1) trivial trusted party, and 2) trivial trusted-storage party, which are explained below.
Trivial trusted party. In this setting, the organizations send their plain data to a trusted party. Next, the trusted party employs the considered functionality in the plaintext scenario on the received data to get the result, and shares it with the authorized parties.
Trivial trusted-storage party. In this setting, we assume that the trusted party has limited storage resources. Therefore, the organizations first encrypt their data, and then sends their encrypted data (instead of plain data) to the storage party (instead of the trusted party). Then, when needed, the trusted party downloads the encrypted data from the storage party, decrypts them, and employs the considered functionality in the plaintext scenario on the downloaded data to get the result. Finally, the trusted party shares the result with the authorized parties.

However, we are looking for a solution that it has different settings for secure data sharing, and aimed to reduce the overheads of the trusted party. In our expected settings, we assume that the trusted party has limited storage and computation capabilities, and only performs the necessary coordinations between parties 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 evaluator that wants to perform an evaluation function, first receives an evaluation key from the trusted party and then downloads the considered encrypted data from the storage party.

Table 1 provides an asymptotic comparison of different solutions for a secure evaluation 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 difference between each solution compared to the other solutions.

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

Overheads	Party type	Trivial trusted party	Trivial trusted-storage party	Our expected solution
Storage	Client	O(nm)	-	-
	Trusted	-	-	-
	Storage	-	O(nm)	O(nm)
	Evaluator	-	-	-
Communication	Client	O(nm)	O(nm)	O(nm)
	Trusted	O(m)	O(nm)	O(1)
	Storage	-	-	-
	Evaluator	-	-	O(nm)
Computation	Client	-	-	-
	Trusted	O(nm)	O(nm)	-
	Storage	-	-	-
	Evaluator	-	-	O(nm)

n: the total number of clients, m: the maximum number of data values.

Functional Encryption (FE) schemes and their various types are the cryptographic tools that meet our requirements for the expected settings. These schemes provide a new paradigm for encryption which extends the traditional "all-or-nothing" requirement of the cryptosystems in a much more flexible way. In the following, we briefly review this flexibility for different types of FE schemes.

Functional Encryption (FE) schemes. A basic FE scheme [5, 17] is a cryptographic primitive that extends the decryption algorithm of the symmetric/asymmetric encryption schemes in a much more flexible way (from all-or-noting decryption of a ciphertext to the computation of a specific function on the original message related to a ciphertext). More precisely, in the basic FE schemes, 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 extracting the original value x from the ciphertext ct).

Multi-Input Functional Encryption (MIFE) schemes. In the basic FE schemes, the function f has only one input component (i.e., f is a 1-ary function). An extension on these schemes is to support the n-ary functions. The MIFE schemes [12] cover such a setting. More precisely, in the MIFE schemes, the decryption algorithm needs a decryption key assigned to an n-ary function f, and n ciphertexts $ct_1, \cdots, ct_n$ computed respectively for values $x_1, \cdots, x_n$ to output $f(x_1, \cdots, x_n)$.

Multi-Client Functional Encryption (MCFE) schemes. A strict subset of the MIFE schemes are called the MCFE schemes [7, 12]. In these schemes, the input components for the n-ary function f are labeled by a tag t (for every time-step), and are independently provided by n distinct clients. More precisely, in this setting, the decryption algorithm requires a decryption key assigned to an n-ary function f, and n ciphertexts $ct_{t,1}, \cdots, ct_{t,n}$ labeled for the same tag t (according to the values $x_{t,1}, \cdots, x_{t,n}$, respectively) to output $f(x_{t,1}, \cdots, x_{t,n})$.

In this paper, we specifically study the set intersection functionality as a function in the FE schemes. In this regard, we consider a more flexible version of the MCFE schemes for set intersection, that we call it Flexible Multi-Client Functional Encryption for set intersection (FMCFE-SI). These schemes support a flexible choice of the clients involved in any set intersection functionality.

Figure 1 summarizes the computable results using an authorized function produced by each of the FE schemes. As it can be seen, the FMCFE schemes have the most flexibility to issue the desired function in addition to having the least peripheral information leakage for each requested function.

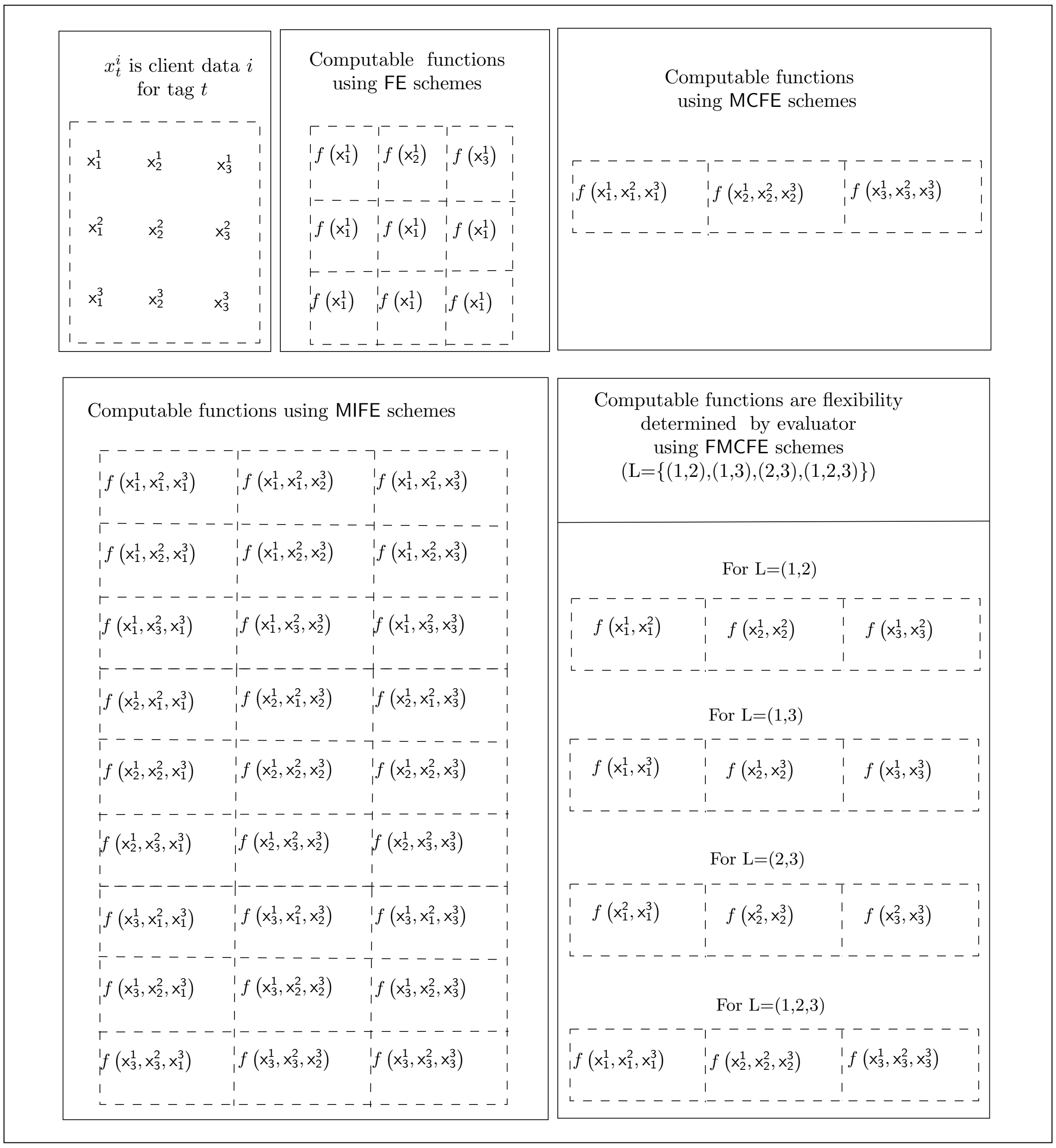

x_t^i is client data i for tag t
x_1^1 x_2^1 x_3^1
x_1^2 x_2^2 x_3^2
x_1^3 x_2^3 x_3^3
Computable functions using FE schemes
$f(x_1^1)$ $f(x_2^1)$ $f(x_3^1)$
$f(x_1^1)$ $f(x_1^1)$ $f(x_1^1)$
$f(x_1^1)$ $f(x_1^1)$ $f(x_1^1)$
Computable functions using MCFE schemes
$f(x_1^1, x_1^2, x_1^3)$ $f(x_2^1, x_2^2, x_2^3)$ $f(x_3^1, x_3^2, x_3^3)$
Computable functions using MIFE schemes
$f(x_1^1, x_1^2, x_1^3)$ $f(x_1^1, x_1^2, x_2^3)$ $f(x_1^1, x_1^2, x_3^3)$
$f(x_1^1, x_2^2, x_1^3)$ $f(x_1^1, x_2^2, x_2^3)$ $f(x_1^1, x_2^2, x_3^3)$
$f(x_1^1, x_3^2, x_1^3)$ $f(x_1^1, x_3^2, x_2^3)$ $f(x_1^1, x_3^2, x_3^3)$
$f(x_2^1, x_1^2, x_1^3)$ $f(x_2^1, x_1^2, x_2^3)$ $f(x_2^1, x_1^2, x_3^3)$
$f(x_2^1, x_2^2, x_1^3)$ $f(x_2^1, x_2^2, x_2^3)$ $f(x_2^1, x_2^2, x_3^3)$
$f(x_2^1, x_3^2, x_1^3)$ $f(x_2^1, x_3^2, x_2^3)$ $f(x_2^1, x_3^2, x_3^3)$
$f(x_3^1, x_1^2, x_1^3)$ $f(x_3^1, x_1^2, x_2^3)$ $f(x_3^1, x_1^2, x_3^3)$
$f(x_3^1, x_2^2, x_1^3)$ $f(x_3^1, x_2^2, x_2^3)$ $f(x_3^1, x_2^2, x_3^3)$
$f(x_3^1, x_3^2, x_1^3)$ $f(x_3^1, x_3^2, x_2^3)$ $f(x_3^1, x_3^2, x_3^3)$
Computable functions are flexibility determined by evaluator using FMCFE schemes (L={(1,2),(1,3),(2,3),(1,2,3)})
For L=(1,2)
$f(x_1^1, x_1^2)$ $f(x_2^1, x_2^2)$ $f(x_3^1, x_3^2)$
For L=(1,3)
$f(x_1^1, x_1^3)$ $f(x_2^1, x_2^3)$ $f(x_3^1, x_3^3)$
For L=(2,3)
$f(x_1^2, x_1^3)$ $f(x_2^2, x_2^3)$ $f(x_3^2, x_3^3)$
For L=(1,2,3)
$f(x_1^1, x_1^2, x_1^3)$ $f(x_2^1, x_2^2, x_2^3)$ $f(x_3^1, x_3^2, x_3^3)$

Fig. 1: The computable functions using different types of FE schemes

1.1 Contributions

The MCFE schemes proposed to support the set intersection have the following weaknesses:

- For each new time-step or new evaluator, the set elements of each client must be available to compute a set intersection functionality, and so clients must be online.
- The decryption key given to the evaluator is applicable to all sets and not an arbitrary subset of them. As a result, this case causes more information to be leaked.

In this paper, we address both weaknesses by introducing a more flexible version of the MCFE schemes for the set intersection (termed FMCFE-SI) that supports flexible choice of the clients involved in any set intersection functionality, and also clients can securely share their sets on a public bulletin board. To this end, we redefine the syntax and security notions of the MCFE schemes for our new scheme. We also consider, based on corruption of clients, three variant security definitions for our scheme: 1) dynamic corruptions, 2) static corruptions, and 3) passive corruption. See Section 5.2 for more details on each security notion.

In literature, the computation of set intersection in polynomial-time complexity (for n sets with an arbitrary polynomial size) such that no information is revealed except the intersection result, is known as an open problem [14]. Also, Kamp et al. [13] investigated this problem from a theoretical perspective. However, we propose an FMCFE-SI construction that computes the set intersection in polynomial-time, but for a relaxed version (see Section 6). In the relaxed version, we assume that the size of our universal set is polynomial in the security parameter. We show that our construction, for an universal set of size m and the client sets with maximum size n, computes the set intersection in time $\mathcal{O}(nm)$ (see Section 7.4). We also prove that our construction has adaptive indistinguishability-based security under passive corruption. Our proof relies on the Symmetric eXternal Diffie-Hellman (SXDH) assumption in the standard model (see Section 7.1).

We leave it as future work to propose the FMCFE-SI schemes under static and dynamic corruptions. Another natural open problem is to propose the FMCFE-SI scheme in polynomial-time complexity in original setting (without constraint on size of the universal set).

1.2 Paper organization

The remaining of this paper is organized as follows. We discuss related work in Section 2. The problem statement and the preliminaries are explained in Section 3 and Section 4, respectively. We formalize the syntax and security of the FMCFE-SI schemes in Section 5. Our proposed FMCFE-SI construction, and its security and performance analysis are presented in Section 6 and Section 7, respectively. Finally, Section 8 concludes the paper and points out future directions.

2 Related work

Recently, Kamp et al. [13] proposed several multi-client functional encryption schemes for set intersection functionality 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 from the clients, can learn the output of considered operation on all sets without having to learn the plaintext set of each individual client. The constructions proposed in [13], for set intersection and cardinality, are in both multi-client setting ($n \geq 2$) and 2-client setting ($n = 2$). However, both constructions support only a set operation functionality, 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 [15] 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 the server. Next, the server upon receiving a decryption key for a pair of sets, can evaluate the set intersection for this pair[2].

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

[2] It should be noted that the construction proposed in [15], despite being provided for multi-client settings, 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.

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

– The evaluation of set intersection must be done non-interactively. It should be noted that this feature is provided by FE schemes, and therefore the schemes described above also have this feature.
– The evaluation of set intersection must be done in polynomial-time complexity, while no information is revealed except the intersection result. This problem in [14] and [13] is investigated from a theoretical point of view.
– 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 such that the evaluator can perform their evaluation at any time (without any online client).
– The last feature is the flexibility to compute the functions with arbitrary arity. In previous works, the decryption key given to the evaluator is applicable to all sets and not an arbitrary subset of them.

Table 2 summarizes a detailed overview of the discussed related works.

Table 2: A detailed overview of the different schemes for set intersection

Construction	#Client	Public Sharing	Intersection Flexibility	Corruption	Constraint	Tool	Assumption	Security Model
Kamp et al. [13]	$n = 2$	✗	-	-	-	Group-based	DDH	Standard
Lee et al. [15]	$n = 2$	✓	-	-	-	Pairing-based	Variant of XDH	Random Oracle
Kamp et al. [13]	$n \geq 2$	✗	✗	Static	-	Group-based	DDH	Random Oracle
Our construction	$n \geq 2$	✓	✓	Passive	$\|\mathcal{W}\| = \mathsf{poly}(\lambda)$	Pairing-based	SXDH	Standard

n: the total number of clients, m: the maximum size of each set, $\mathcal{W}$: the universal set of all possible values available to clients.

2.1 Some similar schemes

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

Private Set Intersection (PSI) Protocols. The classic problem of Private Set Intersection (PSI) protocols in the standard Multi-Party Computation (MPC) [10] are somewhat similar to MCFE schemes for set intersection. The basic scenario model for these protocols includes several parties who each hold a private set locally. 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 [9]. This scenario model considers a new party compared to previous model (termed provider). In this model, the parties outsource their sets to the provider 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 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 evaluated set intersection, while in the MCFE schemes we require a dedicated evaluator to only learn this outcome.

Multi-adjustable Join (M-Adjoin) Schemes. The M-Adjoin scheme, first proposed by Khazaei and Rafiee [14], 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 join result to the user.
The similarities and differences. In here columns play the role of sets, and join queries play the role of requested 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. It is worth mentioning that these schemes can provide the ability to perform flexible queries for every arbitrary subset of columns.

3 System model and problem statement

In this section, we describe the system model, threat model, and problem statement for our proposed scheme.

3.1 System model

Our system 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 functionality 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 considered 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 of clients, and provides the required information to compute the set intersection for the evaluator.

Remark 3.1 *In our model, each of the parties introduced above can play the role of evaluator. For example, the evaluator and storage provider responsibilities can be performed by one party, simultaneously. This case is very similar to cloud scenario models where computation and storage are outsourced to an external provider. Even if these responsibilities are considered separately, they are similar to cloud scenarios where storage service is assigned to one provider and computation service are assigned to one or more providers.*

3.2 Threat model

In our scenario model, we consider that the clients, the evaluators, and storage provider are honest-but-curious. We say that a party is the honest-but-curious if follow the scheme correctly, but plays the role of an eavesdropper to infer additional information from encrypted sets, requested set intersections and corresponding responses.

3.3 Problem statement

Let $\mathcal{W}$ is the universal set of all possible values available to the clients, and $s_i \subseteq \mathcal{W}$ is the set belong to i-th client. Also, suppose that we have n clients, that each of them hold a private set $s_i \subseteq \mathcal{W}$. Our problem is to design a scheme, 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 their sets. Figure 2 summarizes the system model and the problem statement of our scheme.

4 Preliminaries

In this section, we introduce some notations and basic cryptographic primitives that are used throughout the paper. The readers familiar with these concepts can safely skip this section.

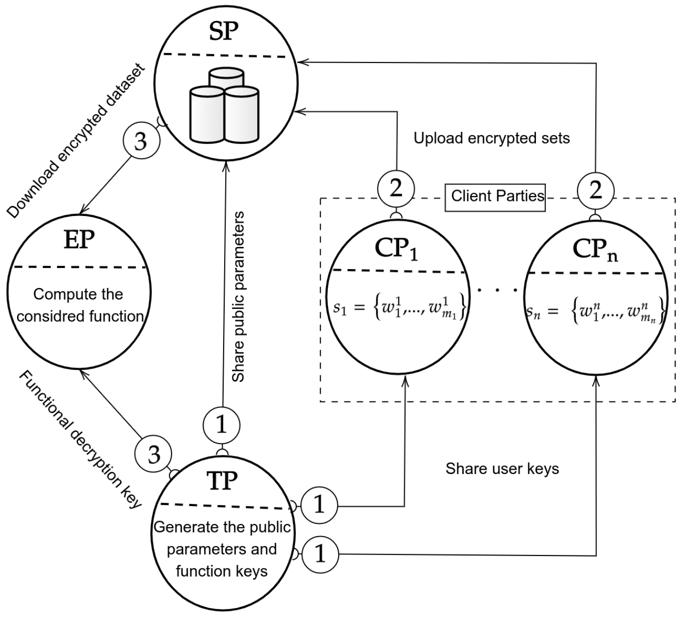

SP
EP
Compute the considred function
TP
Generate the public parameters and function keys
CP$_1$
CP$_n$
$s_1 = \{w_1^1, \ldots, w_{m_1}^1\}$
$s_n = \{w_1^n, \ldots, w_{m_n}^n\}$
Client Parties
Upload encrypted sets
Download encrypted dataset
Share public parameters
Functional decryption key
Share user keys
1
2
3

Fig. 2: An overview of the system model and problem statement of our scheme

4.1 Notation

Throughout the paper, we consider the symbol λ to denote the security parameter. We use $[n]$ to denote the set $\{0, 1, \cdots, n\}$, 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 (i.e., $010|101 = 010101$).

4.2 Basic primitives

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

$$\begin{aligned}&|\Pr[k \leftarrow \{0,1\}^\lambda : \mathcal{A}^{\mathsf{F}_k(\cdot)}(1^\lambda) = 1] - \\ &\Pr[f \leftarrow \mathsf{RF} : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda),\end{aligned}$$

where RF is the set of all the functions from X to Y.
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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, 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 a tuple $Param = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$.

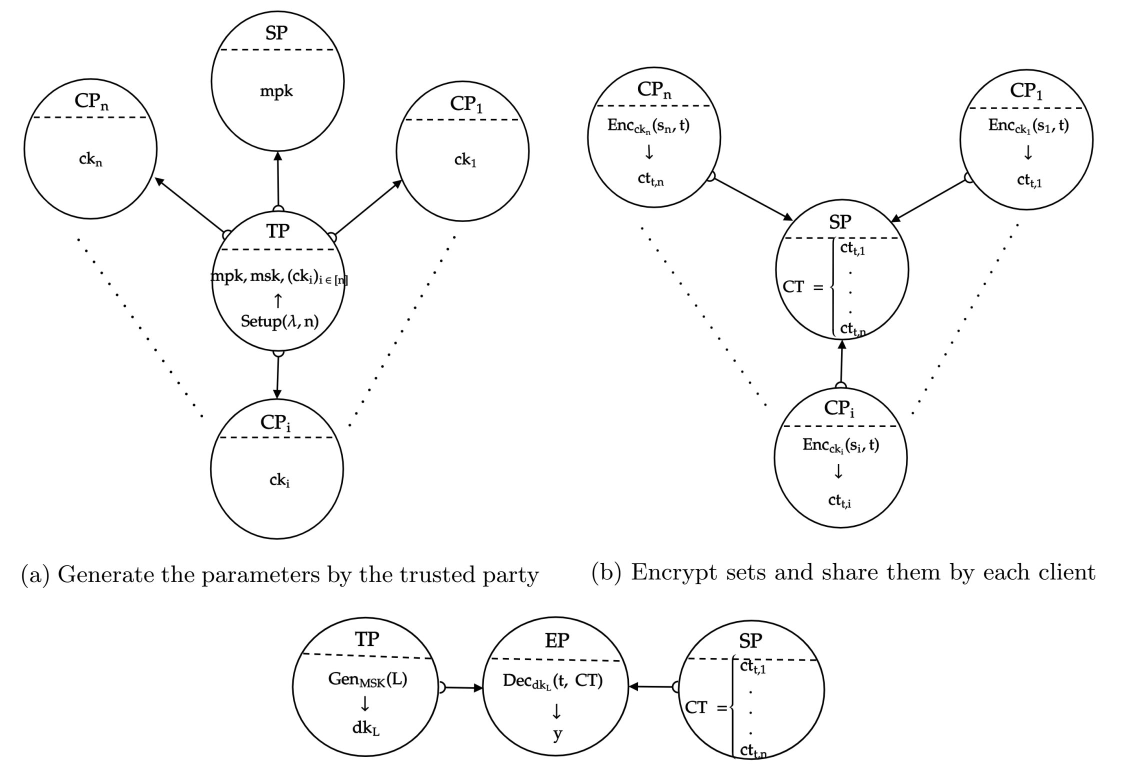
5 FMCFE-SI scheme

In this section, we define the syntax and security notions of Flexible Multi-Client Functional Encryption (FMCFE-SI) schemes. It should be noted that we define the FMCFE-SI schemes in private-key setting only, 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.

5.1 FMCFE-SI syntax

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

For n clients, the FMCFE-SI schemes are used as follows. At first, the trusted party generates a set of public parameters mpk, a master secret key msk, and a list of user 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 mpk on the public bulletin board. Next, each client encrypts its set using an encryption algorithm denoted by Enc, and sends it to the storage provider. Later, when the trusted party wants to send a functional decryption key to the evaluator, he 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. Figure 3 summarizes the process of how to use our scheme.



(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: The process of how to use our scheme

In the following, we provide a formal syntax definition of our FMCFE-SI scheme.

Definition 5.1 (FMCFE-SI **syntax**) *A flexible multi-client functional encryption scheme for set intersection is a collection of four polynomial-time algorithms* $\Pi =$ (Setup, Enc, Keygen, Dec) *such that:*

– $\big(mpk, msk, (ck_i)_{i=1}^n\big) \leftarrow \mathsf{Setup}(\lambda, n)$*: takes as input a security parameter* λ *and a pre-determined number of the clients* n*, and returns a set of public parameters* mpk*, a master secret key* msk *and a list of client keys* $(ck_i)_{i=1}^n$.
– $ct_{t,i} \leftarrow \mathsf{Enc}(ck_i, s_i, t)$*: takes as input the client key* ck_i*, a set* s_i *and a tag* t*, and returns a ciphertext* $ct_{t,i}$.
– $dk_L \leftarrow \mathsf{Keygen}(msk, L)$*: takes as input the master secret key* msk *and a list of client identifiers* L*, and outputs a functional decryption key* dk_L.
– $y \leftarrow \mathsf{Dec}(dk_L, t, CT)$*: takes as input a functional 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 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:*

$$\Pr\begin{bmatrix} \big(mpk, msk, (ck_i)_{i=1}^n\big) \leftarrow \mathsf{Setup}(\lambda, n); \\ \forall i \in [L] \quad ct_{t,i} \leftarrow \mathsf{Enc}(ck_i, s_i, t); \\ dk_L \leftarrow \mathsf{Keygen}(msk, L) : \\ \mathsf{Dec}(dk_L, t, CT) = \bigcap_{i\in L} s_i \end{bmatrix} = 1 \ .$$

5.2 FMCFE-SI security

In this subsection, we formalize the security notions of FMCFE-SI schemes using indistinguishability-based security games. To this end, we use the security definition ideas proposed in [1] to handle the flexibility of the function arity, and also we use the security definition ideas presented in [7] to handle the time-step in our FMCFE-SI schemes.

In our security game, we need that the encrypted values of the clients do not reveal any information about the plaintext values. 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. In addition, in our game we consider capabilities for the adversary such as: the adaptive query to get the evaluation key for any desired subset, and adaptive query to encrypt any value for any client in any time-step. In this game, we also consider the constraints that cause the adversary to easily and trivially 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 $\mathsf{Exp}^{\mathsf{aIND}}_{\mathcal{A},\Pi}(\lambda)$**:**

1. **Initialization phase:** The challenger runs $\big(mpk, msk, (ck_i)_{i=1}^n\big) \leftarrow \mathsf{Setup}(\lambda, n)$, 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 $\mathsf{Enc}(\cdot,\cdot,\cdot)$, $\mathsf{Keygen}(\cdot)$ and $\mathsf{Corrupt}(\cdot)$ queries, which are defined as follows:
   (a) $\mathsf{Enc}(i, s_i, t)$**:** The challenger computes and returns to the adversary $\mathcal{A}$ a ciphertext $ct_{t,i} \leftarrow \mathsf{Enc}(ck_i, s_i, t)$. For any given pair (i, t), only one query is allowed and later queries involving the same pair (i, t) are ignored.
   (b) $\mathsf{Keygen}(L)$**:** The challenger runs $dk_L \leftarrow \mathsf{Keygen}(msk, L)$, and returns to the adversary the decryption key dk_L.
   (c) $\mathsf{Corrupt}(i)$**:** The challenger adds i to $\mathcal{CS}$ (i.e., $\mathcal{CS} = \mathcal{CS} \cup \{i\}$), removes i from the set $\mathcal{HS}$ (i.e., $\mathcal{HS} = \mathcal{HS}\backslash\{i\}$), 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 $\mathsf{Enc}(i, s^*_{i,0}, s^*_{i,1}, t^*)$, and as a response obtains a ciphertext $ct_{t^*,i} \leftarrow \mathsf{Enc}(ck_i, s^*_{i,b}, t^*)$. 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 $\mathsf{Exp}^{\mathsf{aIND}}_{\mathcal{A},\Pi}(\lambda)$, if for every security parameter λ, in all transcripts of the game $\mathsf{Exp}^{\mathsf{aIND}}_{\mathcal{A},\Pi}(\lambda)$, it holds that for every queried list $L = (i_1, \cdots, i_l)$, there does not exist two sequences $(s^*_{i_1,0}, \cdots, s^*_{i_l,0})$ and $(s^*_{i_1,1}, \cdots, s^*_{i_l,1})$ for tag t^* such that:

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

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 $\mathsf{Enc}(i, s^{'}, s^{''}, t^*)$ such that $s^*_{i,0} = s^{'}$ and $s^*_{i,1} = s^{''}$.

Definition 5.2 (**FMCFE-SI security**) *A flexible multi-client functional encryption scheme for set intersection such as $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Keygen}, \mathsf{Dec})$ is* aIND*-secure if for every PPT valid adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

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

Weaker security notions. We can also consider two weaker security notions for our scheme:

- Passive security (P-aIND): No corruption queries (Corrupt) are issued in the game $\mathsf{Exp}^{\mathsf{aIND}}_{\mathcal{A},\Pi}$ (i.e., $\mathcal{CS} = \emptyset$).
- Static security (S-aIND): The corruption queries (Corrupt) are sent before the initialization phase in the game $\mathsf{Exp}^{\mathsf{aIND}}_{\mathcal{A},\Pi}$.

6 Our FMCFE-SI construction

In this section, we propose an FMCFE-SI construction that supports a universal set of polynomial size (in the security parameter λ), and in Section 7.1 we prove that it is aIND-secure under passive corruption. 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, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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 use a pseudo-random function $\mathsf{F} : \{0,1\}^\lambda \times \{0,1\}^{2\lambda} \to \mathbb{G}_1$. The algorithms of our FMCFE-SI construction are defined as follows:

– $\big(\mathbf{mpk}, \mathbf{msk}, (\mathbf{ck_i})^{\mathbf{n}}_{\mathbf{i=1}}\big) \leftarrow \mathsf{Setup}(\lambda, \mathbf{n})$: On input of the security parameter λ and a pre-determined number of clients n, acts as follows:
  1. Run bilinear map generator $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e) \leftarrow \mathcal{G}(\lambda)$,
  2. Choose a word-key $wk \in \{0,1\}^\lambda$ uniformly at random,
  3. For every $i \in [n]$, sample k_i and define $ck_i = (wk, k_i)$,
  4. Set the master secret key $msk = (ck_i)^n_{i=1}$, and the public parameters $mpk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T,$ $g_1, g_2, q, e)$.

– $\mathbf{ct_{t,i}} \leftarrow \mathsf{Enc}(\mathbf{ck_i}, \mathbf{s_i}, \mathbf{t})$: Given the client key ck_i, a set s_i, and a tag t, it works as follows:
  1. Parse ck_i as (wk, k_i),
  2. For every $w \in \mathcal{W}$, compute $ct_{t,i}[w]$ as follows:
     - if $w \in s_i$, then set $ct_{t,i}[w] = \mathsf{F}_{wk}(w|t)^{k_i} \in \mathbb{G}_1$,
     - if $w \notin s_i$, then select a random value $r \leftarrow \mathbb{Z}^*_q$ and set $ct_{t,i}[w] = g_1^r \in \mathbb{G}_1$.

– $dk_L \leftarrow \mathsf{Keygen}(msk, L)$: On input of the master secret key msk and a list of clients L, compute dk_L as follows:
  1. Parse msk as $\big(ck_i = (wk, k_i)\big)^n_{i=1}$ and L as $(i_1, \cdots, i_l)$,
  2. For every $i \in L$, sample $z_i \leftarrow \mathbb{Z}^*_q$ such that $\sum\limits_{i\in L} z_i = 0$,
  3. For every $i \in L$, compute $at_i \leftarrow g_2^{z_i k_i^{-1}}$, and finally define $dk_L = (at_i)_{i\in L}$.

– $y \leftarrow \mathsf{Dec}(dk_L, t, CT)$: Given the decryption key dk_L, a tag t, and a $|L|$-list ciphertext CT, compute y as follows:
  1. Parse 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 append w to y, for every $w \in \mathcal{W}$.

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:

$$\prod_{i \in L} e(ct_{t,i}[w], at_i) = \prod_{i \in L} e(g_1^{\hat{w}_{t,i} k_i}, g_2^{z_i k_i^{-1}}) = \prod_{i \in L} e(g_1, g_2)^{\hat{w}_{t,i} z_i} \tag{6.1}$$

where $\left(mpk, msk, (ck_i)_{i=1}^{n}\right)$ is the output of $\mathsf{Setup}(\lambda, n)$, the ciphertext $ct_{t,i}$ is the output of $\mathsf{Enc}(ck_i, s_i, t)$ for any $i \in L$, the decryption key $dk_L = (at_i)_{i \in L}$ is the output of $\mathsf{Keygen}(msk, L)$, 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 Equation 6.1 we have:

$$\prod_{i \in L} e(g_1, g_2)^{\hat{w}_{t,i} z_i} = e(g_1, g_2)^{\sum_{i \in L} \hat{w}_{t,i} z_i} = e(g_1, g_2)^{\hat{w}_{t,i} \sum_{i \in L} z_i}, \tag{6.2}$$

and since $\sum_{i \in L} z_i = 0$, we have $e(g_1, g_2)^{\hat{w}_{t,i} \sum_{i \in L} z_i} = 1$, 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 $\mathsf{F}_{wk}(w, t)^{k_i} \neq g_1^r$, where r is a random value from $\mathbb{Z}_q^*$. It is easy to show that the probability that $e(g_1, g_2)^{\hat{w}_{t,i} \sum_{i \in L} z_i} = 1$ (i.e., $w \in \mathsf{Dec}(dk_L, t, CT)$) is at most $\frac{1}{q} + \mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda)$ is some negligible function.

$\square$

7 Security and performance analysis

In this section, we first prove that our $\mathsf{FMCFE\text{-}SI}$ construction of Section 6 is $\mathsf{P\text{-}aIND}$-secure. Then, we provide an asymptotic comparison between the schemes presented in Section 2. We also provide a concrete evaluation of the effective components of the different algorithms in our construction.

7.1 Security analysis

In this subsection, we first review a well-known computational hardness assumption that it is used to prove the security of our $\mathsf{FMCFE\text{-}SI}$ construction. Then, we prove the security of our construction in accordance with the security definition given in Section 5.2 under passive corruption.

7.2 Computational hardness assumption

The Symmetric External Diffie-Hellman (SXDH) assumption, formalized in [18, 3, 11, 4], is a computational hardness assumption that underlies the security of several pairing-based cryptosystems such as [2, 6]. Our $\mathsf{FMCFE\text{-}SI}$ construction is proved secure under a standard variant of the SXDH assumption.

Assumption 7.1 (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 7.1 is Decisional Diffie-Hellman assumption in G2 (denoted by DDH2), which is identical to 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 $\mathbb{G}_2$.

7.3 Security proof

In this subsection, we prove that the proposed $\mathsf{FMCFE\text{-}SI}$ construction of Section 6 is $\mathsf{P\text{-}aIND}$-secure.

Theorem 7.2 *If F be a secure pseudo-random function, and the DDH1 assumption holds relative to $\mathcal{G}$, then our $\mathsf{FMCFE\text{-}SI}$ construction of Section 6 is $\mathsf{P\text{-}aIND}$-secure.*

Proof. We need to show that advantage of every PPT valid adversary $\mathcal{A}$ in the game $\mathsf{Exp}^{\mathsf{P\text{-}aIND}}_{\mathcal{A},\Pi}(\lambda)$ is negligible. Let $\mathsf{Exp}^{\mathsf{P\text{-}aIND}}_{\mathcal{A},\$F}(\lambda)$ denotes the game obtained from $\mathsf{Exp}^{\mathsf{P\text{-}aIND}}_{\mathcal{A},\Pi}(\lambda)$ by replacing the pseudo-random function F_{wk} with truly random function f. By the pseudo-randomness property of F_{wk}, it holds that advantage of the valid adversary $\mathcal{A}$ in distinguish between $\mathsf{Exp}^{\mathsf{P\text{-}aIND}}_{\mathcal{A},\Pi}(\lambda)$ and $\mathsf{Exp}^{\mathsf{P\text{-}aIND}}_{\mathcal{A},\$F}(\lambda)$ is negligible. Therefore, to prove $\mathsf{P\text{-}aIND}$ security of the scheme Π, it is sufficient to show that the advantage of every PPT valid adversary $\mathcal{A}$ in the game $\mathsf{Exp}^{\mathsf{P\text{-}aIND}}_{\mathcal{A},\$F}(\lambda)$ is negligible.

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

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

$$f(t,w) = \begin{cases} (g_1^m)^{f_{t,w}} & \text{if } t = t^* \\ g_1^{f_{t,w}} & \text{o.w.} \end{cases}, \tag{7.1}$$

$$k_i = a \cdot a_i^{'}, \tag{7.2}$$

where $f_{t,w}$ and $(a_i^{'})_{i\in[n]}$ are the random values from $\mathbb{Z}_q^*$. 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 $mpk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$.
  2. For every $i \in [n]$, the challenger samples $a_i^{'}$ from $\mathbb{Z}_q^*$, and defines $ck_i = (f, a \cdot a_i^{'})$.
  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:
  - $\mathsf{Enc}(\mathbf{i}, \mathbf{s_i}, \mathbf{t})$: 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 $\mathcal{A}$.
  - $\mathsf{Keygen}(\mathbf{L})$: 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 $r_i = a^{-1} z_i$.
    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 $\mathsf{Enc}(i, s_{i,0}^*, s_{i,1}^*, t^*)$, 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 $\mathcal{A}$.

  Note that if $s = am$, the ciphertext is distributed properly according the scheme, and if $s = z$ then the challenger returns a ciphertext of a randomly distributed set element.
- **Post-challenge query phase.** The challenger replies to the adversary's queries similar to the pre-challenge phase.

□

7.4 Performance analysis

In this subsection, we first present an asymptotic comparison of the schemes presented in Section 2, and then provide a concrete evaluation of our scheme in the terms of time and space complexity.

Table 3 shows an asymptotic comparison of the schemes presented in Section 2, for computing set intersection in a time-step, in terms of the total number of the clients, the storage and communication overheads of various parties (for n clients that each hold a set of size m), and the intersection time (for n sets each of size m). For each of these schemes, according to the scenario models described, the storage and computation overheads are clear. However, in the following we only investigate the asymptotic time complexity for these schemes.

The set intersection method used in [13] and [15] for 2-client setting. Let S_1 and S_2 be two sets with cardinality m_1 and m_2, respectively. Also, assume that $m_1 \geq m_2$. We can use a hash table H as follows. We first store all of the elements of set m_2 in H. Then, for every element $w \in m_1$, we investigate if w is in H. If the answer is yes, we append w to our result set. Therefore, the time complexity for these schemes can be achieved in $\mathcal{O}(m_1)$.

The set intersection method used in [13] for multi-client setting. Suppose that $S_1, \cdots, S_n$ be n sets, each of size m. For each combination of set elements, i.e.; $(w_1, \cdots, w_n) \in S_1 \times \cdots \times S_n)$, we must investigate the equality of these elements. Therefore, the required time complexity to do this is $\mathcal{O}(m^n)$.

Our set intersection method for multi-client setting. Similar to the previous case, we have sets $S_1, \cdots, S_n$ each of size m, except that m is the size of the universal set (instead of the size of the set). As mentioned before, we consider the set intersection problem for multiple clients for a relaxed version in which the size of the universal set is polynomial in the security parameter. In our settings, the size of each set is extend to the size of the universal set. To evaluate the set intersection, for every element of the universal set, we need to check whether this element is in all client sets. Therefore, the required time complexity to do this is $\mathcal{O}(mn)$.

Remark 7.3 *It should be noted that using our scheme ideas, we can straightforwardly modify the scheme proposed in [13] for multiple clients ($n \geq 2$) in such a way that it solves the relaxed set intersection problem in polynomial time. However, this new scheme still provides the possibility to evaluate the set intersection for n sets and not any arbitrary subset.*

Table 3: An asymptotic comparison for a time-step

Construction	#Client	Storage				Communication				Intersection Time
		Client	Evaluator	Storage Provider	Central Party	Client	Evaluator	Storage Provider	Central Party	
Kamp et al. [13]	$n = 2$	$2m$		0	$\mathcal{O}(1)$	$2m$		0	$\mathcal{O}(1)$	$\mathcal{O}(m)$
Lee et al. [15]	$n = 2$	0	$2m$			0	$2m$			
Kamp et al. [13]	$n \geq 2$	mn		0	$n + \mathcal{O}(1)$	mn		0	$n + \mathcal{O}(1)$	$\mathcal{O}(m^n)$
Our construction	$n \geq 2$	0	mn			0	mn			$\mathcal{O}(mn)$

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

Since that these schemes have different security notions compared to our scheme, and also our scheme is based on bilinear maps (which has a moderate overhead compared to other cryptographic primitives), our goals of performance analysis are:

- Finding a good view of the execution time of the effective components of the different algorithms in our scheme,
- Highlighting the execution time of the set intersection to solve the problem in the original and relaxed cases.

We implemented our scheme in Java, and used the jPBC library [8] 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 4 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), the bilinear map computation in the decryption algorithm, and the equality test in the decryption algorithm (for any two elements in $\mathbb{G}_T$).

Table 4: A concrete evaluation of our construction.

Algorithm	Setup (Setup)	Encryption (Enc) (for every set element)	Key generation (Keygen) (for every selected set)	Adjust (map computation in Dec)	Compare (for any two elements in $\mathbb{G}_T$)
Time (milliseconds)	1124	0.4	3	3	2×10^{-4}
Size (byte)	979	40	120	120	240

In the following, based on the execution times given in Table 4 and for a specific setting, we investigate a concrete efficiency analysis of the set intersection problem in both original and relaxed versions. Remember that in the relaxed version we assumed that the size of universal set $\mathcal{W}$ is polynomial in the security parameter.

Suppose we have 5 clients that each of them hold a set of size 1000. In such a setting, for a relaxed version using our FMCFE-SI scheme we need to spend approximately 15 seconds while for the original version using the scheme proposed in [13] it is approximately 2×10^8 seconds (about 6 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 MCFE-SI scheme, where an evaluator can learn the outcome of the set intersection for every arbitrary subset of a pre-determined number of clients (instead of all clients). In this regard, we formalized syntax and security notions of the FMCFE-SI schemes, and proposed a construction for a relaxed version of the set intersection that satisfies these notions. Additionally, we also showed that for practical use of the FMCFE-SI schemes, this relaxation is necessary.

Future contributions can be made in aspects such as: introducing the FMCFE-SI schemes that satisfy security notions under static and dynamic corruptions, developing the FMCFE-SI schemes for a decentralized setting in which the trusted party is removed and the clients work together to generate the decryption keys, and providing the FMCFE-SI schemes with polynomial-time complexity for the original problem and not the relaxed version.

References

1. Agrawal, S., Clear, M., Frieder, O., Garg, S., O'Neill, A., Thaler, J.: Ad hoc multi-input functional encryption. In: 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
2. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable RFID tags via insubvertible encryption. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11. pp. 92–101. ACM (2005). https://doi.org/10.1145/1102120.1102134, https://doi.org/10.1145/1102120.1102134
3. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. IACR Cryptology ePrint Archive **2005**, 417 (2005), http://eprint.iacr.org/2005/417
4. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19. pp. 41–55. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_3, https://doi.org/10.1007/978-3-540-28628-8_3
5. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011)
6. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26. pp. 302–321. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11426639_18, https://doi.org/10.1007/11426639_18
7. Chotard, J., Sans, E.D., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 703–732. Springer (2018)
8. De Caro, A., Iovino, V.: jpbc: Java pairing based cryptography. In: 2011 IEEE symposium on computers and communications (ISCC). pp. 850–855. IEEE (2011)
9. Dong, C., Chen, L., Camenisch, J., Russello, G.: Fair private set intersection with a semi-trusted arbiter. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 128–144. Springer (2013)

10. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: International conference on the theory and applications of cryptographic techniques. pp. 1–19. Springer (2004)
11. Galbraith, S.D., Rotger, V.: Easy decision diffie-hellman groups. LMS Journal of Computation and Mathematics **7**, 201–218 (2004)
12. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 578–602. Springer (2014)
13. van de Kamp, T., Stritzl, D., Jonker, W., Peter, A.: Two-client and multi-client functional encryption for set intersection. In: Australasian Conference on Information Security and Privacy. pp. 97–115. Springer (2019)
14. Khazaei, S., Rafiee, M.: Towards more secure constructions of adjustable join schemes. IEEE Transactions on Dependable and Secure Computing (2020)
15. Lee, K., Seo, M.: Functional encryption for set intersection in the multi-client setting. IACR Cryptol. ePrint Arch. **2020**, 1154 (2020)
16. Rafiee, M., Khazaei, S.: Security of multi-adjustable join schemes: Separations and implications. IEEE Transactions on Dependable and Secure Computing (2021)
17. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 457–473. Springer (2005)
18. Scott, M.: Authenticated id-based key exchange and remote log-in with simple token and PIN number. IACR Cryptology ePrint Archive **2002**, 164 (2002), http://eprint.iacr.org/2002/164