Functional Encryption for Set Intersection in the Multi-Client Setting

Kwangsu Lee* Minhye Seo†

Abstract

Functional encryption for set intersection (FE-SI) in the multi-client environment is that each client i encrypts a set X_i associated with time T by using its own encryption key and uploads it to a cloud server, and then the cloud server which receives a function key of the client indexes i, j from a trusted center can compute the intersection $X_i \cap X_j$ of the two client ciphertexts. In this paper, we first newly define the concept of FE-SI suitable for the multi-client setting. Then, we propose an efficient FE-SI scheme in asymmetric bilinear groups and prove the static security of our scheme under newly introduced assumptions. In our FE-SI scheme, a ciphertext consists of $O(\ell)$ group elements, a function key consists of a single group element, and the decryption algorithm has $O(\ell^2)$ complexity where ℓ is the size of a set in the ciphertext. Next, we propose another FE-SI scheme with time-constrained keys that limits the ability of function keys to be valid only for a specified time period T, and proves the static security of our scheme. Finally, we prove that the two assumptions hold in the general group model to provide confidence in the two newly introduced assumptions.

Keywords: Functional encryption, Private set intersection, Multi-client setting, Bilinear maps, Contact tracing.

*Sejong University, Seoul, Korea. Email: kwangsu@sejong.ac.kr.

†Duksung Women's University, Seoul, Korea. Email: mhseo@duksung.ac.kr.

Contents

1 Introduction

Functional encryption (FE) is a new extension of public-key encryption [11]. In FE, a ciphertext is associated with a message x and a private key is associated with a function f, and the decryption algorithm reveals the result of function calculation $f(x)$ [10]. An FE scheme that supports an arbitrary function f can be constructed by using indistinguishability obfuscation [21]. However, to date, designing an efficient construction for indistinguishability obfuscation is a difficult problem. Another way to implement an efficient FE scheme is to limit the expressiveness of the functions supported by FE. Recently, a new FE scheme that supports the inner product operations of ciphertexts and private keys was introduced [3, 5]. By using the FE scheme for inner-products, it is possible to build an efficient FE scheme that performs statistical operations such as average and weighted sum.

It is an interesting research direction to devise other efficient FE scheme that provides new functions that cannot be expressed using inner products. In this paper, we pay attention to a function that calculate the intersection of two sets. Many studies have been conducted on private set intersection (PSI) that computes the intersection of two sets without revealing anything else [16, 18, 19, 26]. Basically, PSI is a two-party protocol in which two parties with sets X and Y exchange encrypted messages multiple times with each other to compute the intersection $X \cap Y$. At this time, the two parties cannot obtain information on the other's set items except the intersection items. Many PSI protocols have been proposed, but fundamentally it has a disadvantage that requires interactions between parties. In an environment where a large number of users stores their encrypted data in a cloud server and do not access the cloud server afterwards, an FE scheme for set intersection that does not require interactions is more suitable than a PSI protocol that requires interactions between users. An FE scheme for set intersection can be constructed by using a multi-input FE scheme for arbitrary functions, but this approach is still ineffective because this FE scheme requires indistinguishability obfuscation [24].

Recently, FE schemes that support set intersection operations was proposed by Kamp et al. [35]. They defined the concept of multi-client FE for set intersection by extending the concept of multi-client FE introduced in [24]. In addition, they proposed a two-client FE scheme for set intersection that operates between two clients and a multi-client FE scheme for set intersection that operates between multiple clients. However, their FE definition for set intersection has a problem of lacking the flexibility to control the set intersection operation because only one function key is set at the setup stage. For this reason, their FE scheme has a problem that the setup algorithm needs to be performed for each pair of clients to perform the set intersection. For example, if there are n clients, a maximum of n^2 setups is required for the set intersection between two clients and each client has to store n^2 encryption keys.

A better FE scheme for set intersection is that only one setup is performed even if there are n clients, and each client must own only one encryption key. In addition, this FE scheme can issue many function keys for set intersection, and an entity with a function key must be able to freely calculate the set intersection on two ciphertexts of the corresponding clients. In this paper, we ask whether it is possible to efficiently construct this better FE scheme for set intersection.

1.1 Our Contributions

In this paper, we define FE that supports the set intersection operation and propose two FE schemes for set intersection in bilinear groups.

Definition. First, we define functional encryption for set intersection (FE-SI) that supports basic intersection operation. In FE-SI, a trusted center provides an independent encryption key EK_i for each client i and

generates a function key $SK_{i,j}$ for the intersection operation on two clients i, j. Each client with an index i creates a ciphertext C_i on time T by encrypting a set X_i by using its encryption key. After that, a third party with a function key $SK_{i,j}$ can compute the intersection $X_i \cap X_j$ from two ciphertexts C_i and C_j of two clients i, j by running the decryption algorithm when the ciphertexts are generated at the same time T. We modify the definition of FE-SI to define functional encryption for set intersection with time-constrained keys (FE-SI-TCK) that issues a function key that is valid only for a limited time period T. The function key of FE-SI-TCK can limit the life-time of the function key because the function key is additionally associated with time and it is valid only for ciphertexts of the same time.

FE for Set Intersection. In order to design an FE-SI scheme in asymmetric bilinear groups, we devise a method to derive a temporal key TK to be used for encryption and decryption when the ciphertext elements of two clients encrypt the same set item x. That is, when the encryption keys of clients i and j are given as (α_i, β_i) and (α_j, β_j) respectively, the ciphertext elements of two clients i, j are formed as $H(T\|x)^{\alpha_i}, H(T\|x)^{\alpha_j}$ where H is a hash function and x is a set item, and a function key for indexes i, j is provided as $SK_{i,j} = \hat{g}^{\beta_i/(\alpha_i+\alpha_j)}$. Then a temporal key is derived as $e(H(T\|x)^{\alpha_i} \cdot H(T\|x)^{\alpha_j}, SK_{i,j}) = e(H(T\|x), \hat{g})^{\beta_i}$ by using a pairing operation. In our FE-SI scheme, a function key is composed of a single group element and the size of a ciphertext is proportional to the number of set items. The decryption algorithm requires $O(\ell^2)$ complexity where ℓ is the size of a set because it needs to try all possible pairs of ciphertext elements of two clients. In order to prove the security of our FE-SI scheme, we define a static-IND security model such that an attacker initially submits all corrupted clients, challenge sets, and all function key queries. To prove the static-IND security of our FE-SI scheme, we newly introduce two dynamic assumptions derived from the FE-SI scheme and show that these assumptions hold in the generic group model. In addition, we present the extensions of our FE-SI scheme that support associated data encryption, set intersection cardinality, and multi-party set intersection.

FE-SI with Time-Constrained Keys. The function key for indexes i, j of our FE-SI scheme is very powerful because it can be used to compute the set intersection of all ciphertexts of two clients i, j. A way to provide additional control to the FE-SI scheme is to limit the availability of the function key to a specific time period. By modifying our FE-SI scheme, we propose an FE-SI-TCK scheme that has the ability to issue a function key that is only valid for a time period T. In order to limit the function key for the time T, we derive a time encryption key $(\alpha_{i,T}, \beta_{i,T})$ for each individual time period T from an original encryption key EK_i and use this key for ciphertext encryption on time T. That is, by using the pseudo-random function PRF, we calculate $\alpha_{i,T} = PRF(z, 1\|T)$ and $\beta_{i,T} = PRF(z, 2\|T)$ where z is the original encryption key. In this case, the function key is generated as $SK_{i,j,T} = \hat{g}^{\beta_{i,T}/(\alpha_{i,T}+\alpha_{j,T})}$ for the time T, which is only valid for the time T. In order to prove security of our FE-SI-TCK scheme, we argue that our FE-SI-TCK scheme is static-IND secure if our FE-SI scheme is also static-IND secure. In addition, the FE-SI-TCK scheme can be easily extended to support a time-range function key that is valid from time T_L to time T_R, and to provide forward secrecy that protects past ciphertexts even when the encryption key of a client is exposed.

1.2 Application

Privacy-Preserving Contact Tracing. Due to the worldwide spread of infectious diseases such as the coronavirus, there is a need for a way to trace people who have come into contact with a confirmed patient for public health [1]. To this end, people want to check whether a common location exists between their visited locations and the visited locations of the confirmed patient while hiding their locations. One method is to use the existing private set intersection (PSI) protocol to calculate the set intersection. However, this method is difficult to directly apply since additional interactions between users are needed to calculate the

intersection and the load of client computation is relatively high. Recently, private set intersection cardinality (PSI-CA) protocols for contact tracing has been proposed [17, 34]. Another way is to use an MCFE scheme for set intersection. To this end, a hospital cloud server stores the encrypted data of the visited locations of the confirmed patient. Then, when a user encrypts his visited locations and uploads them to the cloud server, the hospital cloud server receives a function key for set intersection cardinality and calculates the set intersection cardinality between the user and the confirmed person. If the cardinality of the set intersection is positive, then the cloud server notifies the user that there is a possibility of contact. After that, the user may additionally receive a function key for set intersection and decrypts the exact intersection locations. Another advantage of using the MCFE scheme is that the set intersection operation of ciphertexts corresponding to the same time period is allowed, but the set intersection operation of ciphertexts in different time periods is not allowed.

1.3 Related Work

Functional Encryption. The concept of functional encryption (FE) was introduced by Boneh et al. [10, 11]. Identity-based encryption, hierarchical identity-based encryption, attribute-based encryption, and predicate encryption, which are widely known, can all be viewed as special forms of FE [9, 23, 25, 29]. In terms of performing computation on encrypted data, FE is similar to homomorphic encryption (HE), but there is a difference that unlike the ciphertext of the computation $f(x)$ is the output of the evaluation algorithm in HE, the computation $f(x)$ itself is the output of the decryption algorithm in FE [22]. It is known that an FE scheme that supports all polynomial-size circuits can be designed by using indistinguishability obfuscation [21]. By expanding the concept of FE, the concepts of multi-input FE (MI-FE) and multi-client FE (MC-FE) were introduced, and MI-FE and MC-FE schemes can be also designed with indistinguishability obfuscation [24]. For efficient FE schemes, functional encryption for inner-products (FE-IP) was introduced and many FE-IP schemes were proposed [3, 5]. Since then, the FE-IP scheme has been extended to the multi-input FE-IP and multi-client FE-IP schemes [4, 15, 31]. An efficient FE scheme for quadratic functions has also been proposed, which has more expressive power than inner product operations [6]. Recently, an MC-FE scheme that supports conjunctive equality and range queries has been proposed [30].

Private Set Intersection. The private set intersection (PSI) protocol was introduced by Freedman et al. [19]. The PSI protocol is a protocol that calculates the intersection $X \cap Y$ when two parties A, B each own sets X, Y respectively, and does not expose other set information other than the set intersection of X and Y. Secure PSI protocols can be designed by using various cryptographic tools such as Diffie-Hellman key exchange [27], oblivious polynomial evaluation [19], oblivious pseudo-random functions [18], garbled circuits [26], oblivious transfer protocols [16], and fully homomorphic encryption [14]. Basically, the PSI protocol is an interactive one that exchanges multiple messages between two parties. An interesting variant of the PSI protocol is the outsourced PSI protocol using a cloud server [28]. In the outsourcing PSI protocol, since the cloud server performs the operations performed by individual clients instead, it is possible to reduce conversations and operations between clients. However, this outsourcing PSI protocol also requires interactions between clients during initial setup or at some stage.

Public-Key Searchable Encryption. Public-key encryption with keyword search (PEKS) is a kind of public-key encryption that allows to search keywords in ciphertexts by using a trapdoor generated by a trusted party [8]. In PEKS, ciphertext and trapdoor are associated with each keyword, and the test algorithm can check whether the ciphertext keyword and the trapdoor keyword are the same or not. By using a PEKS scheme, we can construct a special kind of the set intersection protocol [12]. That is, one client generates ciphertexts by using a public key where each ciphertext is associated with a keyword $x \in X$, and the other

client receives trapdoors for a set Y from the trusted party where each keyword is associated with a keyword $y \in Y$. In this case, the client who has trapdoors can calculate the intersection $X \cap Y$ by running the test algorithm for each pair of ciphertexts and trapdoors. In general, a PEKS scheme can be constructed by using an anonymous IBE scheme [2].

2 Preliminaries

In this section, we define functional encryption, symmetric-key encryption, and pseudo-random function. We also introduce complexity assumptions to prove the security of our FE-SI schemes proposed in this paper.

2.1 Notation

Let n be a positive integer. The notation $[n]$ is defined as a set $\{1,\ldots,n\}$, and the notation $[n_1,n_2]$ is defined as a set $\{n_1,\ldots,n_2\}$. Given two strings a and b, $a\|b$ is the concatenation of the two strings.

2.2 Functional Encryption

Functional encryption is an extension of public key encryption that outputs the computation on encrypted data $f(x)$ instead of outputting the original message x in the decryption process in which a ciphertext is associated with a message x, and a private key is associated with a function f [10]. Functional encryption can be modified in various ways and can be extended to multiple-input functional encryption that processes multiple ciphertexts, and multi-client functional encryption that processes ciphertext generated by multiple clients that have independent encryption keys [24]. The following is the syntax of public key functional encryption.

Definition 2.1 (Functional Encryption)**.** A functional encryption (FE) scheme consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

Setup(1^λ). The setup algorithm takes as input a security parameter λ. It outputs a master key MK and public parameters PP.

GenKey(f,MK,PP). The key generation algorithm takes as input a function f, the master key MK, and the public parameters PP. It outputs a secret key SK_f.

Encrypt(x,PP). The encryption algorithm takes as input a message x and the public parameters PP. It outputs a ciphertext CT.

Decrypt(CT,SK_f,PP). The decryption algorithm takes as input a ciphertext CT encrypting a message x, a secret key SK_f corresponding to a function f, and the public parameters PP. It outputs a value $f(x)$.

The correctness property of FE is defined as follows: For all $(MK,PP) \leftarrow$ **Setup**(1^λ), $SK_f \leftarrow$ **GenKey**(f,MK,PP) for any function $f \in \mathcal{F}$, and $CT \leftarrow$ **Encrypt**(x,PP) for any $x \in \mathcal{X}$, it is required that **Decrypt**$(CT,SK_f,PP) = f(x)$.

2.3 Symmetric Key Encryption

Symmetric-key encryption is encryption that uses the same secret key for encryption and decryption algorithms. The detailed syntax of the symmetric-key encryption is described as follows.

Definition 2.2 (Symmetric Key Encryption)**.** A symmetric key encryption (SKE) scheme consists of three algorithms **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

GenKey(1^λ). The key generation algorithm takes as input a security parameter λ. It outputs a symmetric key K.

Encrypt(M,K). The encryption algorithm takes as input a message $M \in \mathcal{M}$ and the symmetric key K. It outputs a ciphertext C.

Decrypt(C,K). The decryption algorithm takes as input a ciphertext CT and the symmetric key K. It outputs a message M or a symbol $\perp$.

The correctness property of SKE is defined as follows: For all K generated by **GenKey** and any message $M \in \mathcal{M}$, it is required that **Decrypt**(**Encrypt**$(M,K),K) = M$.

The standard security model of symmetric-key encryption is indistinguishability security against chosen-plaintext attacks (IND-CPA), but we define one-message security as a weaker form of security. One-message security is to provide only one challenge ciphertext to an attacker, and the IND-CPA security guarantees one-message security.

Definition 2.3 (One-Message Security)**.** The one-message security of SKE is defined in the following experiment $\mathbf{EXP}^{SKE}_{\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. **Setup**: $\mathcal{C}$ generates a secret key K by running **GenKey**(1^λ). It keeps K to itself.
2. **Challenge**: $\mathcal{A}$ submits challenge messages M_0^*, M_1^* where $|M_0^*| = |M_1^*|$. $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and gives a challenge ciphertext CT^* to $\mathcal{A}$ by running **Encrypt**(M_μ^*, K).
3. **Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$ of μ. $\mathcal{C}$ outputs 1 if $\mu = \mu'$ or 0 otherwise.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{SKE}_{\mathcal{A}}(\lambda) = \left| \Pr[\mathbf{EXP}^{SKE}_{\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right|$. An SKE scheme is one-message secure if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

2.4 Pseudo-Random Function

A pseudo-random function (PRF) is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ where $\mathcal{K}$ is a key space, $\mathcal{X}$ is a domain, and $\mathcal{Y}$ is a codomain. Let $F(k,\cdot)$ be an oracle for a uniformly chosen $k \in \mathcal{K}$ and $f(\cdot)$ be an oracle for a uniformly chosen function $f : \mathcal{X} \to \mathcal{Y}$. We say that a PRF F is secure if for all efficient adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as $\mathbf{Adv}^{PRF}_{\mathcal{A}}(\lambda) = \left| \Pr[\mathcal{A}^{F(k,\cdot)} = 1] - \Pr[\mathcal{A}^{f(\cdot)} = 1] \right|$ is negligible in the security parameter λ.

2.5 Bilinear Groups

A bilinear group generator $\mathcal{G}$ takes as input a security parameter λ and outputs a tuple $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ where p is a random prime and $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ are three cyclic groups of prime order p. Let g and $\hat{g}$ be generators of $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively. The bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u \in \mathbb{G}, \forall \hat{v} \in \hat{\mathbb{G}}$ and $\forall a, b \in \mathbb{Z}_p$, $e(u^a, \hat{v}^b) = e(u, \hat{v})^{ab}$.

2. Non-degeneracy: $\exists g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}}$ such that $e(g, \hat{g})$ has order p in $\mathbb{G}_T$.

We say that $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are asymmetric bilinear groups with no efficiently computable isomorphisms if the group operations in $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable, but there are no efficiently computable isomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$.

2.6 Complexity Assumptions

In order to prove the security of the proposed FE-SI schemes, we introduce two new complexity assumptions. These two assumptions are not static assumptions that are composed of fixed group elements, but dynamic assumptions in which the group elements of the assumptions change by given parameters.

The first assumption is an assumption derived from our FE-SI scheme. This is a modification of the widely known external Diffie-Hellman (XDH) assumption, and this assumption says that it is difficult to distinguish the Diffie-Hellman tuple even if additional group elements related to Q and J parameters are provided. The second assumption is also an assumption derived from our FE-SI scheme.

Let n be a positive integer, ρ be a target index such that $\rho \in [n]$, and $Q = \{(i, j)\}$ be a set of index pairs such that $i, j \in [n]$ and $i < j$. From n, ρ, and Q, we define an index set $J = \{k : 1 \leq k \neq \rho \leq n$ such that $(k, \rho) \notin Q$ if $k < \rho$ and $(\rho, k) \notin Q$ if $k > \rho\}$. This set can be computed by using the function *ComputeJ* which is described as follows:

$\underline{ComputeJ(n, \rho, Q)}$ where $Q = \{(i, j)\}$
1. Initialize J as the empty set.
2. For each $k \in \{1, \ldots, n\} \setminus \{\rho\}$:
If $k < \rho$ and $(k, \rho) \notin Q$, then add k to J.
If $k > \rho$ and $(\rho, k) \notin Q$, then add k to J.
3. Output J.

For example, if we let $n = 4$, $\rho = 2$, and $Q = \{(1,4), (2,3), (2,4)\}$, then we obtain $J = \{1\}$ since $(1,2) \notin Q$, $(2,3) \in Q$, and $(2,4) \in Q$.

Assumption 1. Let $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear group randomly generated by $\mathcal{G}(1^\lambda)$. Let $g, \hat{g}$ be random generators of $\mathbb{G}, \hat{\mathbb{G}}$ respectively. Let n, ρ, Q, J be defined above. The Assumption 1 for (n, ρ, Q, J) is that if the challenge tuple

$$D = \big((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e),\ g,\ g^a,\ \{g^{b_i}\}_{i=1}^n,\ \{g^{ab_k}\}_{k \in J},\ \hat{g},\ \{\hat{g}^{1/(b_i + b_j)}\}_{(i,j) \in Q}\big) \text{ and } Z$$

are given, no probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = g^{ab_\rho}$ from $Z = Z_1 = g^d$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{\text{A1-}(n,\rho,Q,J)}(\lambda) = \big|\Pr[\mathcal{A}(D, Z_0) = 0] - \Pr[\mathcal{A}(D, Z_1) = 0]\big|$ where the probability is taken over random choices of $a, b_1, \ldots, b_n, d \in \mathbb{Z}_p$.

Assumption 2. Let $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear group randomly generated by $\mathcal{G}(1^\lambda)$. Let $g, \hat{g}$ be random generators of $\mathbb{G}, \hat{\mathbb{G}}$ respectively. Let n, ρ, Q be defined above. The Assumption 2 for (n, ρ, Q) is that if the challenge tuple

$$\begin{aligned} D = \big((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e),\ g,\ g^a,\ \{g^{b_i}\}_{i=1}^n,\ \{g^{ab_k}\}_{1 \leq k \neq \rho \leq n},\\ \hat{g},\ \{\hat{g}^{c_i}\}_{1 \leq i \neq \rho \leq n},\ e(g, \hat{g})^{c_\rho},\ \{\hat{g}^{c_i/(b_i + b_j)}\}_{(i,j) \in Q}\big) \text{ and } Z \end{aligned}$$

are given, no probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = e(g,\hat{g})^{ac_\rho}$ from $Z = Z_1 = e(g,\hat{g})^{d}$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{A2\text{-}(n,\rho,Q)}(\lambda) = \left|\Pr[\mathcal{A}(D,Z_0)=0] - \Pr[\mathcal{A}(D,Z_1)=0]\right|$ where the probability is taken over random choices of $a, b_1, \ldots, b_n$, $c_1, \ldots, c_n, d \in \mathbb{Z}_p$.

We analyzed that both of these new assumptions hold in the generic group model proposed by Shoup [33] in Section 5.

3 Functional Encryption for Set Intersection

In this section, we first introduce functional encryption for set intersection (FE-SI) and define the security model of FE-SI. Next, we construct an FE-SI scheme using a bilinear map and analyze the security of our FE-SI scheme using the cryptographic assumptions of the previous section.

3.1 Definition

To define functional encryption for set intersection, we modify the definition of public key functional encryption to consider a multi-client setting in which individual clients own individual encryption keys [24]. First, a trusted center creates a master key, an encryption key for each client, and public parameters by running the setup algorithm, and the individual encryption keys are delivered securely to clients. After that, a client generates a ciphertext for a set X_i associated with time T by using their own encryption key. If a third party who wants to perform the set intersection operation obtains a function key for client indexes i, j from the trusted center, and computes the set intersection of the ciphertexts generated by two clients i and j at time T by running the decryption algorithm. A more detailed syntax of FE-SI is given as follows.

Definition 3.1 (Functional Encryption for Set Intersection)**.** A functional encryption for set intersection (FE-SI) scheme for $\mathcal{D}$ and $\mathcal{T}$ consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

Setup$(1^\lambda, n)$. The setup algorithm takes as input a security parameter λ and the number of clients n. It outputs a master key MK, client encryption keys $\{EK_i\}_{i=1}^n$, and public parameters PP.

GenKey(i, j, MK, PP). The key generation algorithm takes as input two client indexes $i, j \in [n]$ such that $i < j$, the master key MK, and public parameters PP. It outputs a function key $SK_{i,j}$.

Encrypt(X_i, T, EK_i, PP). The encryption algorithm takes as input a set $X_i = \{x_{i,1}, \ldots, x_{i,\ell_i}\}$ where $x_{i,j} \in \mathcal{D}$, a time period $T \in \mathcal{T}$, the client encryption key EK_i, and the public parameters PP. It outputs a ciphertext $CT_{i,T}$.

Decrypt$(CT_{i,T}, CT_{j,T}, SK_{i,j}, PP)$. The decryption algorithm takes as input two ciphertexts $CT_{i,T}$ and $CT_{j,T}$ for the same time T, a function key $SK_{i,j}$, and the public parameters PP. It outputs a set $X_i \cap X_j$ where X_i and X_j are associated with $CT_{i,T}$ and $CT_{j,T}$ respectively.

The correctness property of FE-SI is defined as follows: For all $MK, \{EK_i\}, PP \leftarrow$ **Setup**$(1^\lambda, n)$, any $SK_{i,j} \leftarrow$ **GenKey**(i, j, MK, PP), and all $CT_{i,T} \leftarrow$ **Encrypt**(X_i, T, EK_i, PP) and $CT_{j,T} \leftarrow$ **Encrypt**(X_j, T, EK_j, PP) for any X_i, X_j and the same time T, it is required that

- **Decrypt**$(CT_{i,T}, CT_{j,T}, SK_{i,j}, PP) = X_i \cap X_j$ except with negligible probability.

To define the security model of FE-SI, we modify the security model of multi-client functional encryption (MC-FE) defined by Goldwasser et al. [24]. For the security model of FE-SI, we consider a static security model in which an attacker pre-specifies information related to the attack target and function key queries. Initially, the attacker specifies the list of corrupted clients, the challenge target sets, the challenge time period, and the set of function key queries. At this time, the challenge target sets and the set of function key queries submitted by the attacker are restricted so that the challenge target sets cannot be easily identified by using the function keys in order to prevent trivial attacks. After that, the attacker obtains the encryption keys of the corrupted clients and the challenge ciphertexts for the challenge sets. Additionally, the attacker can request the previously specified function keys and ciphertexts for a time period other than the challenge time period. Finally, the attacker identifies the challenge set of the challenge ciphertexts. A more detailed definition of the static security model of FE-SI is defined as follows.

We first define a function $CIQ(\{X_k\}, Q)$ for a group of item sets $\{X_k\}$ and a set $Q = \{(i, j)\}$ that computes the collected intersection of X_i and X_j for each $(i, j) \in Q$ as follows:

$\underline{CIQ(\{X_k\}_{k\in I}, Q)}$ where $Q = \{(i, j)\}$
1. For each $i \in I$, initialize E_i as the empty set.
2. For each $(i, j) \in Q$,
   calculate $Y = X_i \cap X_j$ and add Y to E_i and E_j respectively.
3. Output $\{E_i\}_{i\in I}$.

Definition 3.2 (Static-IND Security)**.** The static-IND security of FE-SI with corruptions is defined in the following experiment $\mathbf{EXP}^{ST\text{-}IND}_{FE\text{-}SI,\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. **Init**: $\mathcal{A}$ initially submits an index set $\bar{I} \subset [n]$ of corrupted clients. Let $I = \{1, \ldots, n\} \setminus \bar{I}$ be the index set of uncorrupted clients. $\mathcal{A}$ also submits two challenge sets of item sets $\{X^*_{0,k}\}_{k\in I}$, $\{X^*_{1,k}\}_{k\in I}$, a challenge time period T^*, and a set $Q = \{(i, j)\}$ of function key queries with the restriction that $i, j \in I$ for each $(i, j) \in Q$ and $CIQ(\{X^*_{0,k}\}_{k\in I}, Q) = CIQ(\{X^*_{1,k}\}_{k\in I}, Q)$.

2. **Setup**: $\mathcal{C}$ generates a master key MK, encryption keys $\{EK_i\}^n_{i=1}$, and public parameters PP by running **Setup**$(1^\lambda, n)$. It keeps MK and $\{EK_i\}_{i\in I}$ to itself and gives $\{EK_i\}_{i\in\bar{I}}$ and PP to $\mathcal{A}$.

3. **Challenge**: $\mathcal{C}$ flips a random coin $\mu \in \{0, 1\}$ and obtains a ciphertext CT_{i,T^*} by running **Encrypt**$(X^*_{\mu,i}$, $T^*, EK_i, PP)$ for each $i \in I$. $\mathcal{C}$ gives the challenge ciphertexts $\{CT_{i,T^*}\}_{i\in I}$ to $\mathcal{A}$

4. **Query**: $\mathcal{A}$ requests function keys and ciphertexts. $\mathcal{C}$ handles these queries as follows:

   - If this is a function key query for indexes $i, j \in I$ with the restriction that $(i, j) \in Q$, then $\mathcal{C}$ gives a function key $SK_{i,j}$ to $\mathcal{A}$ by running **GenKey**(i, j, MK, PP).
   - If this is a ciphertext query for a client index k, an item set X_k, and a time period T with the restriction that $k \in I$ and $T \neq T^*$, then $\mathcal{C}$ gives a ciphertext $CT_{k,T}$ to $\mathcal{A}$ by running **Encrypt**(X_k, T, EK_k, PP).

5. **Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0, 1\}$ of μ. $\mathcal{C}$ outputs 1 if $\mu = \mu'$ or 0 otherwise.

An FE-SI scheme is static-IND secure with corruptions if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as $\mathbf{Adv}^{ST\text{-}IND}_{FE\text{-}SI,\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{EXP}^{ST\text{-}IND}_{FE\text{-}SI,\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}\right|$ is negligible in the security parameter λ.

3.2 Design Principle

The basic idea of supporting set intersection operations is to implement a private equality test. If a hash function $H : \{0,1\}^* \to \mathbb{G}$ is modeled as a random oracle, a function $H(x)^\alpha$ with a secret α can play the role of pseudo-random function because the output of this function is indistinguishable from a completely random value by the DDH assumption [32]. Additionally, this function supports the private equality test because the output of the function is same if the same input is given. If multiple interactions are allowed between clients, the two clients can compute the set intersection of the two sets X and Y by exchanging $\{H(x)^\alpha\}_{x\in X}$, $\{H(y)^{\alpha'}\}_{y\in Y}$ in the first round and exchanging $\{H(x)^{\alpha\alpha'}\}_{x\in X}$, $\{H(y)^{\alpha'\alpha}\}_{y\in Y}$ in the second round [27].

However, we cannot design a functional encryption scheme for set intersection with the above method because clients are non-interactive in which messages cannot be exchanged between clients in functional encryption. In order to devise a functional encryption scheme that supports the set intersection operation, we have to solve two problems. First, we need to implement a private equality test that non-interactively checks whether the items of two sets are the same. Second, if the items of two sets are the same, we need to implement a method of decrypting the ciphertext of the corresponding item. To this end, we devise an equal-then-derive method in which the correct message decryption key is derived by combining with a function key if the equality of set items is satisfied.

The equal-then-derive method we devised is as follows. The first client generates a ciphertext element $H(x)^{\alpha_1}$ for a set item x using its encryption key α_1. And the second client also generates a ciphertext element $H(y)^{\alpha_2}$ for a set item y using its encryption key α_2. At this time, if the two set items are identical such as $x = y$, we have $H(x)^{\alpha_1} \cdot H(y)^{\alpha_2} = H(x)^{\alpha_1+\alpha_2}$ from the ciphertext elements. Here, if the trusted center provides a function key $\hat{g}^{\beta_1/(\alpha_1+\alpha_2)}$, it is possible to derives a temporal key $TK = e(H(x)^{\alpha_1+\alpha_2}, \hat{g}^{\beta_1/(\alpha_1+\alpha_2)}) = e(H(x), g)^{\beta_1}$ which can be used to encrypt or decrypt a message. To do this, the encryption key of the first client should contain an additional secret β_1, and the first client uses the temporary key TK to create another ciphertext element on a message x by using symmetric-key encryption.

The function $H(x)^\alpha$ previously used for set intersection is pseudo-random function. Therefore, if a client re-encrypts the previously encrypted set item x, then some information of the set item is exposed because the output of this function is the same when the input is the same. To prevent such information leakage, we set a client to encrypt the set associated with a specific time period T, and the set intersection operation to be performed on the ciphertexts of two clients having the same time T. To do this, we use a modified function $H(T\|x)^\alpha$, which contains additional time. In this case, even if the same x is encrypted, the value $H(T\|x)^\alpha$ appears to be a random value when the time T is different. Thus this function is secure since there is no information leakage.

3.3 Construction

Let **SKE = (GenKey**, **Encrypt**, **Decrypt**) be an SKE scheme. An FE-SI scheme is described as follows.

Setup$(1^\lambda, n)$. Let n be the maximum number of clients.

1. It first generates a bilinear group $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ of prime order p with random generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. It chooses two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \mathbb{G}_T \to \{0,1\}^\lambda$.
2. Next, it selects a random PRF key $z \in \{0,1\}^\lambda$ and computes $\alpha_i = PRF(z, 1\|i)$, $\beta_i = PRF(z, 2\|i))$ for each index $i \in [n]$.
3. It outputs a master key $MK = z$, encryption keys $\{EK_i = (\alpha_i, \beta_i)\}_{i=1}^n$ for clients, and public parameters $PP = \big((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e), g, \hat{g}, H_1, H_2\big)$.

GenKey(i, j, MK, PP). Let $i < j$ and $MK = z$. It first computes $\alpha_i = PRF(z, 1\|i)$, $\alpha_j = PRF(z, 1\|j)$, and $\beta_i = PRF(z, 2\|i)$. It outputs a function key $SK_{i,j} = \hat{g}^{\beta_i/(\alpha_i+\alpha_j)}$ by implicitly including i, j.

Encrypt(X_i, T, EK_i, PP). Let $X_i = \{x_{i,1}, \ldots, x_{i,\ell_i}\}$ where $|X_i| = \ell_i$ and $EK_i = (\alpha_i, \beta_i)$.

1. For each index $k \in [\ell_i]$, it proceed as follows: It computes $C_{i,k} = H_1(T\|x_{i,k})^{\alpha_i}$ and derives a temporal key $TK_{i,k} = e(H_1(T\|x_{i,k}), \hat{g})^{\beta_i}$. It obtains $D_{i,k}$ by running **SKE.Encrypt**$(T\|x_{i,k}, H_2(TK_{i,k}))$.
2. It chooses a random permutation π and outputs a ciphertext $CT_{i,T} = \left\{(C_{i,\pi(k)}, D_{i,\pi(k)})\right\}_{k=1}^{\ell_i}$ by implicitly including i, T.

Decrypt$(CT_{i,T}, CT_{j,T'}, SK_{i,j}, PP)$. Let $CT_{i,T} = \{(C_{i,k}, D_{i,k})\}_{k=1}^{\ell_i}$ and $CT_{j,T'} = \{(C_{j,k}, D_{j,k})\}_{k=1}^{\ell_j}$ be ciphertexts such that $i < j$ and $T = T'$. It first initializes a set $Y = \emptyset$.

1. For each index $k_i \in [\ell_i]$ and $k_j \in [\ell_j]$, it proceeds as follows: It computes $TK_{i,k_i} = e(C_{i,k_i} \cdot C_{j,k_j}, SK_{i,j})$ and obtains a message $A\|x$ by running **SKE.Decrypt**$(D_{i,k_i}, H_2(TK_{i,k_i}))$. It adds an item x into Y if $A = T$.
2. It outputs the set Y.

3.4 Correctness

For the correctness of our FE-SI scheme, we need to show that the set intersection of two client ciphertexts can be calculated through the decryption process. Let $CT_{i,T} = \{(C_{i,k}, D_{i,k})\}$ be the ciphertext of a client i, $CT_{j,T} = \{(C_{j,k}, D_{j,k})\}$ be the ciphertext of a client j, and $SK_{i,j}$ be a function key. For the correctness of the decryption process, it is only necessary to show that the correct temporal key TK is derived by combining with a function key if the ciphertext elements from two clients are the encryption on the same item x and these are related to the same time T. The reason is that if the correct temporary key is derived, the message can be decrypted by running the decryption algorithm of symmetric key encryption. If the set items of the two ciphertexts are the same, we can confirm that a temporal key is correctly derived through the following equation

$$e(C_{i,k_i} \cdot C_{j,k_j}, SK_{i,j}) = e(H_1(T\|x_{i,k_i})^{\alpha_i+\alpha_j}, \hat{g}^{\beta_i/(\alpha_i+\alpha_j)}) = e(H_1(T\|x_{i,k_i}), \hat{g})^{\beta_i}.$$

3.5 Security Analysis

To prove the security of the FE-SI scheme, we analyze the security by dividing the case where there is no corrupted client and the case where there are corrupted clients.

In order to prove the static security without corrupted clients of the FE-SI scheme, we devise additional hybrid games that an attacker cannot distinguish between the two challenge sets $\{X^*_{0,k}\}$ and $\{X^*_{1,k}\}$. In the definition of the static security model, the constraint $CIQ(\{X^*_{0,k}\}, Q) = CIQ(\{X^*_{1,k}\}, Q)$ must be satisfied because the attacker can compute the set intersection operation on the challenge set by using function keys. This means that the set items related to the common set $\{E^*_k\} = CIQ(\{X^*_{\mu,k}\}, Q)$ can be computed by the attacker, but the remaining items that do not related to $\{E^*_k\}$ are not revealed to the attacker. Thus even if these ciphertext elements not related to $\{E^*_k\}$ are changed to random values, the attacker can not distinguish this change. Therefore, we define additional hybrid games and change the challenge ciphertext elements that do not related to $\{E^*_k\}$ to random values one by one.

To do this, in the first game, we change the pseudo-random function into a truly random function. In the second game, the ciphertext elements $\{C_{\mu,k}\}$ that do not related to $\{E^*_k\}$ are changed to random values. In

the third game the temporal keys $\{TK_{\mu,k}\}$ that do not related to $\{E_k^*\}$ are changed to random. After that, in the last game, the ciphertext elements $\{D_{\mu,k}\}$ that do not related to $\{E_k^*\}$ are changed to random values. In this last game, the ciphertext elements related to $\{E_k^*\}$ are the same as the original ciphertext, but all other ciphertext elements not related to $\{E_k^*\}$ are changed to random values. Thus the attacker can't tell whether the challenge ciphertext of the last game is related to $\{X_{0,k}^*\}$ or $\{X_{1,k}^*\}$. The details of the security are given as follows.

Theorem 3.1. *The above FE-SI scheme is static-IND secure with no corruptions in the random oracle model if the PRF scheme is secure, the SKE scheme is one-message secure, and the Assumptions 1 and 2 hold.*

Proof. Suppose there exists an adversary that breaks the static-IND security of the FE-SI scheme with no corruptions. We can assume that $I = \{1,\ldots,n\}$ and $\bar{I} = \emptyset$. Let $\{X_{0,1}^*,\ldots,X_{0,n}^*\}$ and $\{X_{1,1}^*,\ldots,X_{1,n}^*\}$ be the challenge sets of item sets where $X_{b,i}^* = \{x_{b,i,1}^*,\ldots,x_{b,i,\ell_i}^*\}$ and $|X_{b,i}^*| = \ell_i$. Let $Q = \{(i,j)\}$ be the set of index pairs related to function key queries. We can derive a set of common sets $\{E_1^*,\ldots,E_n^*\}$ by calling $CIQ(\{X_{0,k}^*\},Q)$ since $CIQ(\{X_{0,k}^*\},Q) = CIQ(\{X_{1,k}^*\},Q)$ by the restriction of the security model. To argue that the adversary cannot win this game, we define a sequence of hybrid games $\mathbf{G}_0,\mathbf{G}_1,\mathbf{G}_2,\mathbf{G}_3$, and $\mathbf{G}_4$. The game $\mathbf{G}_i$ is defined as follows:

Game $\mathbf{G}_0$. The first game $\mathbf{G}_0$ is the original security game defined in Definition 3.2.

Game $\mathbf{G}_1$. In this game $\mathbf{G}_1$, the PRF which is used to generate encryption keys is changed to be truly random function.

Game $\mathbf{G}_2$. This game $\mathbf{G}_2$ is similar to the game $\mathbf{G}_1$ except that the challenge ciphertext components $\{C_{i,k}\}$ are generated as random for all $x_{\mu,i,k}^* \notin E_i^*$.

Game $\mathbf{G}_3$. This game $\mathbf{G}_3$ is slightly changed from the game $\mathbf{G}_2$. That is, the challenge temporal keys $\{TK_{i,k}\}$ are generated as random for all $x_{\mu,i,k}^* \notin E_i^*$.

Game $\mathbf{G}_4$. In the final game $\mathbf{G}_4$, we change the generation of challenge ciphertext components $\{D_{i,k}\}$. That is, the challenge ciphertext components $\{D_{i,k}\}$ are the encryption of random values for all $x_{\mu,i,k}^* \notin E_i^*$. Note that the advantage of the adversary in this game is zero since challenge ciphertext components $\{C_{i,k}\}$ are random and $\{D_{i,k}\}$ are the encryption of random values for all $x_{\mu,i,k}^* \notin E_i^*$.

Let $S_{\mathcal{A}}^{\mathbf{G}_i}$ be the event that an adversary wins in a game $\mathbf{G}_i$. From the following lemmas 3.2, 3.3, 3.4, and 3.5, we obtain the following result

$$\begin{aligned}\mathbf{Adv}_{FE\text{-}SI,\mathcal{A}}^{ST\text{-}IND}(\lambda) \leq& \left|\Pr[S_{\mathcal{A}}^{\mathbf{G}_0}] - \Pr[S_{\mathcal{A}}^{\mathbf{G}_4}]\right| + \Pr[S_{\mathcal{A}}^{\mathbf{G}_4}] \leq \sum_{i=1}^{4}\left|\Pr[S_{\mathcal{A}}^{\mathbf{G}_{i-1}}] - \Pr[S_{\mathcal{A}}^{\mathbf{G}_i}]\right| + \Pr[S_{\mathcal{A}}^{\mathbf{G}_4}] \\ \leq& \mathbf{Adv}_{\mathcal{B}}^{PRF}(\lambda) + n\ell\mathbf{Adv}_{\mathcal{B}}^{A1\text{-}(n,\rho,Q,J)}(\lambda) + n\ell\mathbf{Adv}_{\mathcal{B}}^{A2\text{-}(n,\rho,Q)}(\lambda) + n\ell\mathbf{Adv}_{\mathcal{B}}^{SKE}(\lambda)\end{aligned}$$

where n is the number of clients, ℓ is the maximum size of the challenge item set. This completes our proof. □

Lemma 3.2. *If the PRF is secure, then no polynomial-time adversary can distinguish between $\mathbf{G}_0$ and $\mathbf{G}_1$ with a non-negligible advantage.*

Proof. The proof of this lemma is relatively easy from the security of PRF. That is, we simply change a PRF to a truly random function since there is only one PRF in the FE-SI scheme. We omit the details of this proof. □

Lemma 3.3. *If the Assumption 1 for (n,ρ,Q,J) holds, then no polynomial-time adversary can distinguish between $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ with a non-negligible advantage.*

Proof. To prove this lemma, we additionally define hybrid games $\mathbf{H}_{1,0},\mathbf{H}_{1,1},\ldots,\mathbf{H}_{1,\ell_1},\mathbf{H}_{2,1},\ldots,\mathbf{H}_{i,k},\ldots,\mathbf{H}_{n,\ell_n}$ where $\mathbf{H}_{1,0}=\mathbf{G}_0$ and $\mathbf{H}_{n,\ell_n}=\mathbf{G}_1$. The game $\mathbf{H}_{\rho,\delta}$ is defined as follows:

Game $\mathbf{H}_{\rho,\delta}$. This game $\mathbf{H}_{\rho,\delta}$ is almost identical to the game $\mathbf{G}_1$ except the generation of the components $\{C_{i,k}\}$ in the challenge ciphertexts.

- **Case** $(i<\rho)$ or $(i=\rho\wedge k\leq\delta)$: If $x^*_{\mu,i,k}\in E^*_i$, then the component $C_{i,k}$ is generated as normal. Otherwise ($x^*_{\mu,i,k}\notin E^*_i$), the component $C_{i,k}$ is generated as random.
- **Case** $(i=\rho\wedge k>\delta)$ or $(i>\rho)$: The component $C_{i,k}$ is generated as normal.

Suppose there exists an adversary $\mathcal{A}$ that distinguishes between $\mathbf{H}_{\rho,\delta-1}$ and $\mathbf{H}_{\rho,\delta}$ with a non-negligible advantage. Without loss of generality, we assume that $x^*_{\mu,\rho,\delta}\notin E^*_\rho$ since $\mathbf{H}_{\rho,\delta-1}$ and $\mathbf{H}_{\rho,\delta}$ are equal if $x^*_{\mu,\rho,\delta}\in E^*_\rho$. A simulator $\mathcal{B}$ that solves the Assumption 1 for (n,ρ,Q,J) which will be defined later is described as follows:

Init: $\mathcal{A}$ submits challenge sets $\{X^*_{0,1},\ldots,X^*_{0,n}\}$, $\{X^*_{1,1},\ldots,X^*_{1,n}\}$, a challenge time period T^*, and a set $Q=\{(i,j)\}$ of function key queries. $\mathcal{B}$ proceeds as follows:

1. From n,ρ,Q, it derives an index set J by calling $ComputeJ(n,\rho,Q)$.
2. It receives a challenge tuple $D=((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,g^a,\{g^{b_i}\}_{i=1}^n,\{g^{ab_k}\}_{k\in J},\hat{g},\{\hat{g}^{1/(b_i+b_j)}\}_{(i,j)\in Q})$ and Z of the Assumption 1 for (n,ρ,Q,J) where $Z=g^{ab_\rho}$ or $Z=R\in\mathbb{G}$.
3. It flips a random coin $\mu\in\{0,1\}$ internally and derives a set of common sets $\{E^*_1,\ldots,E^*_n\}$ by calling $CIQ(\{X^*_{\mu,k}\},Q)$.

Setup: $\mathcal{B}$ first chooses random exponents $\beta_1,\ldots,\beta_n\in\mathbb{Z}_p$. Next, it sets $PP=((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,\hat{g},H_1,H_2)$. It prepares a hash table H-list for the H_1 hash function as follows:

1. For each $i\in[n]$ and $k\in[\ell_i]$, it proceeds as follows: If $i\neq\rho$ or $k\neq\delta$, then it selects a random exponent $f_{i,k}\in\mathbb{Z}_p$ and adds $(T^*\|x^*_{\mu,i,k},f_{i,k},g^{f_{i,k}})$ to the H-list. Otherwise, it adds $(T^*\|x^*_{\mu,\rho,\delta},-,g^a)$ to the H-list.

Challenge: $\mathcal{B}$ creates challenge ciphertexts $CT_{1,T^*},\ldots,CT_{n,T^*}$ as follows:

1. For each i and k, it generates ciphertext elements $C_{i,k}$ and $TK_{i,k}$ depending on the following cases:
   - Case $i<\rho$:
     - If $x^*_{\mu,i,k}\in E^*_i$ and $x^*_{\mu,i,k}=x^*_{\mu,\rho,\delta}$, it retrieves $(T^*\|x^*_{\mu,i,k},-,g^a)$ from the H-list, and sets $C_{i,k}=g^{ab_i}$ and creates $TK_{i,k}=e(g^a,\hat{g})^{\beta_i}$. For this case, we show that g^{ab_i} is given in the assumption. If a function key with an index pair (i,ρ) was queried, we have $x^*_{\mu,\rho,\delta}\in E^*_\rho$ by the definition of CIQ. However, we assumed that $x^*_{\mu,\rho,\delta}\notin E^*_\rho$ for this game. Thus a function key with (i,ρ) was not queried and it means that $i\in J$ by the definition of J.
     - If $x^*_{\mu,i,k}\in E^*_i$ and $x^*_{\mu,i,k}\neq x^*_{\mu,\rho,\delta}$, it retrieves $(T^*\|x^*_{\mu,i,k},f_{i,k},g^{f_{i,k}})$ from the H-list, and creates $C_{i,k}=(g^{b_i})^{f_{i,k}}$ and $TK_{i,k}=e(g^{f_{i,k}},\hat{g})^{\beta_i}$.
     - If $x^*_{\mu,i,k}\notin E^*_i$, it retrieves $(T^*\|x^*_{\mu,i,k},f_{i,k},g^{f_{i,k}})$ from the H-list, and chooses a random $C_{i,k}\in\mathbb{G}$ and creates $TK_{i,k}=e(g^{f_{i,k}},\hat{g})^{\beta_i}$.

- Case $i = \rho$:
  - If $k < \delta$ and $x^*_{\mu,\rho,k} \in E^*_\rho$, it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and creates $C_{\rho,k} = (g^{b_\rho})^{f_{\rho,k}}$ and $TK_{\rho,k} = e(g^{f_{\rho,k}}, \hat{g})^{\beta_\rho}$ since $x^*_{\mu,\rho,k} \neq x^*_{\mu,\rho,\delta}$.
  - If $k < \delta$ and $x^*_{\mu,\rho,k} \notin E^*_\rho$, then it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and chooses a random $C_{\rho,k} \in \mathbb{G}$ and creates $TK_{\rho,k} = e(g^{f_{\rho,k}}, \hat{g})^{\beta_\rho}$.
  - If $k = \delta$, it sets $C_{\rho,\delta} = Z$ and creates $TK_{\rho,\delta} = e(g^a, \hat{g})^{\beta_\rho}$ since we assumed that $x^*_{\mu,\rho,\delta} \notin E^*_\rho$.
  - If $k > \delta$, it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and creates $C_{\rho,k} = (g^{b_\rho})^{f_{\rho,k}}$ and $TK_{\rho,k} = e(g^{f_{\rho,k}}, \hat{g})^{\beta_\rho}$ since $x^*_{\mu,\rho,k} \neq x^*_{\mu,\rho,\delta}$.
- Case $i > \rho$:
  - If $x^*_{\mu,i,k} = x^*_{\mu,\rho,\delta}$, it retrieves $(T^* \| x^*_{\mu,i,k}, -, g^a)$ from the H-list, and sets $C_{i,k} = g^{ab_i}$ and creates $TK_{i,k} = e(g^a, \hat{g})^{\beta_i}$. For this case, we show that g^{ab_i} is given in the assumption. If a function key with an index pair (ρ, i) was queried, we have $x^*_{\mu,\rho,\delta} \in E^*_\rho$ by the definition of CIQ. However, we assumed that $x^*_{\mu,\rho,\delta} \notin E^*_\rho$ for this game. Thus a function key with (ρ, i) was not queried and it means that $i \in J$ by the definition of J.
  - If $x^*_{\mu,i,k} \neq x^*_{\mu,\rho,\delta}$, it retrieves $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ from the H-list, and creates $C_{i,k} = (g^{b_i})^{f_{i,k}}$ and $TK_{i,k} = e(g^{f_{i,k}}, \hat{g})^{\beta_i}$.

  Next, it generates a ciphertext element $D_{i,k}$ by running **SKE.Encrypt**$(T^* \| x^*_{\mu,i,k}, TK_{i,k})$

2. It chooses a random permutation π_i and sets a challenge ciphertext $CT_{i,T^*} = \{(C_{i,\pi_i(k)}, D_{i,\pi_i(k)})\}_{k=1}^{\ell_i}$ for each client i.

Query: $\mathcal{A}$ adaptively requests hash, function key, and ciphertext queries. $\mathcal{B}$ handles these queries as follows:

- If this is a hash query for a time period T and an item x, then $\mathcal{B}$ proceeds as follows: If $T \| x$ exists in the H-list, then it retrieves $(T \| x, -, h)$ from H-list and gives h to $\mathcal{A}$. Otherwise, it selects a random exponent $f \in \mathbb{Z}_p$ and adds $(T \| x, f, g^f)$ to the H-list, and then it gives the hash value g^f to $\mathcal{A}$.
- If this is a function key query for indexes i, j such that $(i, j) \in Q$, then $\mathcal{B}$ generates a function key $SK_{i,j} = (\hat{g}^{1/(b_i+b_j)})^{\beta_i}$ since $\hat{g}^{1/(b_i+b_j)}$ is given in the assumption.
- If this is a ciphertext query for a client index i, a set $X_i = \{x_{i,1}, \ldots, x_{i,\ell}\}$, and a time period $T \neq T^*$, then $\mathcal{B}$ generates a ciphertext as follows:
  1. For each $k \in [\ell_i]$, it proceeds as follows: It retrieves $(T \| x_{i,k}, f_k, g^{f_k})$ from the H-list, and sets $C_{i,k} = (g^{b_i})^{f_k}$ and $TK_{i,k} = e(g^{f_k}, \hat{g})^{\beta_i}$. Next, it obtains $D_{i,k}$ by running **SKE.Encrypt**$(T \| x_{i,k}, TK_{i,k})$.
  2. It chooses a random permutation π and creates $CT_{i,T} = \{(C_{i,\pi(k)}, D_{i,\pi(k)})\}_{k=1}^{\ell_i}$.

Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, it outputs 1. Otherwise, it outputs 0.

To complete the proof, we need to analyze the correctness of the simulation. In the case of the challenge ciphertext, ciphertext elements with indexes $i \neq \rho$ or $k \neq \delta$ are all generated correctly by using the values given in the assumption. The challenge ciphertext elements with indexes $i = \rho$ and $k = \delta$ are generated by using the challenge Z of the assumption. If $Z = Z_0 = g^{ab_\rho}$, then it is the same as the game $H_{\rho,\delta-1}$. Otherwise ($Z = Z_1$), it is the same as the game $H_{\rho,\delta}$. The function keys provided to the attacker are easily processed by using the values given in the assumption because function key queries are pre-specified in Q. In the case

of ciphertext such as $T \neq T^*$, all ciphertexts are correctly processed by using the hash table and the values given in the assumption. □

Lemma 3.4. *If the Assumption 2 for (n, ρ, Q) holds, then no polynomial-time adversary can distinguish between $\mathbf{G}_2$ and $\mathbf{G}_3$ with a non-negligible advantage.*

Proof. To prove this lemma, we additionally define hybrid games $\mathbf{H}'_{1,0}, \mathbf{H}'_{1,1}, \ldots, \mathbf{H}'_{1,\ell_1}, \ldots, \mathbf{H}'_{i,k}, \ldots, \mathbf{H}'_{n,\ell_n}$ where $\mathbf{H}'_{1,0} = \mathbf{G}_2$ and $\mathbf{H}'_{n,\ell_n} = \mathbf{G}_3$. The game $\mathbf{H}'_{\rho,\delta}$ is defined as follows:

Game $\mathbf{H}'_{\rho,\delta}$. This game $\mathbf{H}'_{\rho,\delta}$ is almost identical to the game $\mathbf{G}_2$ except the generation of temporal keys $\{TK_{i,k}\}$ in the challenge ciphertexts.

- **Case** $(i < \rho)$ or $(i = \rho \wedge k \leq \delta)$: If $x^*_{\mu,i,k} \in E^*_i$, then the temporal key $TK_{i,k}$ is generated as normal. Otherwise ($x^*_{\mu,i,k} \notin E^*_i$), the temporal key $TK_{i,k}$ is generated as random.
- **Case** $(i = \rho \wedge k > \delta)$ or $(i > \rho)$: The temporal key $TK_{i,k}$ is generated as normal.

Suppose there exists an adversary $\mathcal{A}$ that distinguishes between $\mathbf{H}'_{\rho,\delta-1}$ and $\mathbf{H}'_{\rho,\delta}$ with a non-negligible advantage. Without loss of generality, we assume that $x^*_{\mu,\rho,\delta} \notin E^*_\rho$ since $\mathbf{H}'_{\rho,\delta-1}$ and $\mathbf{H}'_{\rho,\delta}$ are equal if $x^*_{\mu,\rho,\delta} \in E^*_\rho$. A simulator $\mathcal{B}$ that solves the Assumption 2 for (n, ρ, Q) which will be defined later is described as follows:

Init: $\mathcal{A}$ submits challenge sets of item sets $\{X^*_{0,1}, \ldots, X^*_{0,n}\}$, $\{X^*_{1,1}, \ldots, X^*_{1,n}\}$, a challenge time period T^*, and a set $Q = \{(i,j)\}$ of function key queries. $\mathcal{B}$ proceeds as follows:

1. It receives a challenge tuple $D = ((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e), g, g^a, \{g^{b_i}\}^n_{i=1}, \{g^{ab_k}\}_{1 \leq k \neq \rho \leq n}, \hat{g}, \{\hat{g}^{c_i}\}_{1 \leq i \neq \rho \leq n}, e(g,\hat{g})^{c_\rho}, \{\hat{g}^{c_i/(b_i+b_j)}\}_{(i,j) \in Q})$ and Z of the Assumption 2 for (n, ρ, Q) where $Z = e(g,\hat{g})^{ac_\rho}$ or $Z = R \in \mathbb{G}_T$.
2. It flips a random coin $\mu \in \{0,1\}$ internally and derives a set of common sets $\{E^*_1, \ldots, E^*_n\}$ by calling $CIQ(\{X^*_{\mu,k}\}, Q)$.

Setup: $\mathcal{B}$ sets $PP = ((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e), g, \hat{g}, H_1, H_2)$. It prepares a hash table H-list for the H_1 hash function as follows:

1. For each $i \in [n]$ and $k \in [\ell_i]$, it proceeds as follows: If $i \neq \rho$ or $k \neq \delta$, then it selects a random exponent $f_{i,k} \in \mathbb{Z}_p$ and adds $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ to the H-list. Otherwise $(i = \rho \wedge k = \delta)$, it adds $(T^* \| x^*_{\mu,\rho,\delta}, -, g^a)$ to the H-list.

Challenge: $\mathcal{B}$ creates challenge ciphertexts $CT_{1,T^*}, \ldots, CT_{n,T^*}$ as follows:

1. For each i and k, it generates ciphertext elements $C_{i,k}$ and $TK_{i,k}$ depending on the following cases:
   - Case $i < \rho$:
     - If $x^*_{\mu,i,k} \in E^*_i$ and $x^*_{\mu,i,k} = x^*_{\mu,\rho,\delta}$, it retrieves $(T^* \| x^*_{\mu,i,k}, -, g^a)$ from the H-list, and sets $C_{i,k} = g^{ab_i}$ and $TK_{i,k} = e(g^a, \hat{g}^{c_i})$. In this case, g^{ab_i} is given in the assumption since $i \neq \rho$.
     - If $x^*_{\mu,i,k} \in E^*_i$ and $x^*_{\mu,i,k} \neq x^*_{\mu,\rho,\delta}$, it retrieves $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ from the H-list, and sets $C_{i,k} = (g^{b_i})^{f_{i,k}}$ and $TK_{i,k} = e(g^{f_{i,k}}, \hat{g}^{c_i})$.
     - If $x^*_{\mu,i,k} \notin E^*_i$, then it retrieves $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ from the H-list, and selects random $C_{i,k} \in \mathbb{G}$ and $TK_{i,k} \in \mathbb{G}_T$.
   - Case $i = \rho$:

- If $k < \delta$ and $x^*_{\mu,\rho,k} \in E^*_\rho$, it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and sets $C_{\rho,k} = (g^{b_\rho})^{f_{\rho,k}}$ and $TK_{\rho,k} = (e(g,\hat{g})^{c_\rho})^{f_{\rho,k}}$ since $x^*_{\mu,\rho,k} \neq x^*_{\mu,\rho,\delta}$.
- If $k < \delta$ and $x^*_{\mu,\rho,k} \notin E^*_\rho$, then it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and selects random $C_{\rho,k} \in \mathbb{G}$ and random $TK_{\rho,k} \in \mathbb{G}_T$.
- If $k = \delta$, it chooses a random $C_{\rho,\delta} \in \mathbb{G}$ and sets $TK_{\rho,\delta} = Z$ since we assumed that $x^*_{\mu,\rho,\delta} \notin E^*_\rho$.
- If $k > \delta$ and $x^*_{\mu,\rho,k} \in E^*_\rho$, then it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and sets $C_{\rho,k} = (g^{b_\rho})^{f_{\rho,k}}$ and $TK_{\rho,k} = (e(g,\hat{g})^{c_\rho})^{f_{\rho,k}}$ since $x^*_{\mu,\rho,k} \neq x^*_{\mu,\rho,\delta}$.
- If $k > \delta$ and $x^*_{\mu,\rho,k} \notin E^*_\rho$, then it retrieves $(T^* \| x^*_{\mu,\rho,k}, f_{\rho,k}, g^{f_{\rho,k}})$ from the H-list, and selects a random $C_{\rho,k} \in \mathbb{G}$ and creates $TK_{\rho,k} = (e(g,\hat{g})^{c_\rho})^{f_{\rho,k}}$.

- Case $i > \rho$:
  - If $x^*_{\mu,i,k} \in E^*_i$ and $x^*_{\mu,i,k} = x^*_{\mu,\rho,\delta}$, it retrieves $(T^* \| x^*_{\mu,i,k}, -, g^a)$ from the H-list, and sets $C_{i,k} = g^{ab_i}$ and $TK_{i,k} = e(g^a, \hat{g}^{c_i})$. In this case, g^{ab_i} is given in the assumption since $i \neq \rho$.
  - If $x^*_{\mu,i,k} \in E^*_i$ and $x^*_{\mu,i,k} \neq x^*_{\mu,\rho,\delta}$, it retrieves $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ from the H-list, and sets $C_{i,k} = (g^{b_i})^{f_{i,k}}$ and $TK_{i,k} = e(g^{f_{i,k}}, \hat{g}^{c_i})$.
  - If $x^*_{\mu,i,k} \notin E^*_i$, it retrieves $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ from the H-list, and selects a random $C_{i,k} \in \mathbb{G}$ and creates $TK_{i,k} = e(g^{f_{i,k}}, \hat{g}^{c_i})$.

  Next, it generates a ciphertext element $D_{i,k}$ by running **SKE.Encrypt**$(T^* \| x^*_{\mu,i,k}, TK_{i,k})$

2. It chooses a random permutation π_i and sets a challenge ciphertext $CT_{i,T^*} = \{(C_{i,\pi_i(k)}, D_{i,\pi_i(k)})\}_{k=1}^{\ell_i}$ for each client i.

Query: $\mathcal{A}$ adaptively requests hash, token, and ciphertext queries. $\mathcal{B}$ handles these queries as follows:

- If this is a hash query for a time period T and an item x, then $\mathcal{B}$ proceeds as follows: If $T \| x$ exists in the H-list, then it retrieves $(T \| x, -, h)$ from H-list and gives h to $\mathcal{A}$. Otherwise, it selects a random exponent $f \in \mathbb{Z}_p$ and adds $(T \| x, f, g^f)$ to the H-list, and then it gives the hash value g^f to $\mathcal{A}$.

- If this is a function key query for indexes i, j such that $(i, j) \in Q$, then $\mathcal{B}$ generates a function key $SK_{i,j} = \hat{g}^{c_i/(b_i+b_j)}$ since it is given in the assumption.

- If this is a ciphertext query for a client index i, a set $X_i = \{x_{i,1}, \ldots, x_{i,\ell}\}$, and a time period $T \neq T^*$, then $\mathcal{B}$ generates a ciphertext as follows:

  1. For each $k \in [\ell_i]$, it proceeds as follows: It retrieves $(T \| x_{i,k}, f_k, g^{f_k})$ from the H-list and sets $C_{i,k} = (g^{b_i})^{f_k}$. Next, it sets $TK_{i,k} = (e(g,\hat{g})^{c_\rho})^{f_k}$ if $i = \rho$, and it sets $TK_{i,k} = e(g^{f_k}, \hat{g}^{c_i})$ if $i \neq \rho$. It obtains $D_{i,k}$ by running **SKE.Encrypt**$(T \| x_{i,k}, TK_{i,k})$.
  2. It chooses a random permutation π and creates $CT_{i,T} = \{(C_{i,\pi(k)}, D_{i,\pi(k)})\}_{k=1}^{\ell_i}$.

Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, it outputs 1. Otherwise, it outputs 0. □

Lemma 3.5. *If the SKE scheme is one-message secure, then no polynomial-time adversary can distinguish between* $\boldsymbol{G}_3$ *and* $\boldsymbol{G}_4$ *with a non-negligible advantage.*

Proof. To prove this lemma, we additionally define hybrid games $\mathbf{H}''_{1,0}, \mathbf{H}''_{1,1}, \ldots, \mathbf{H}''_{1,\ell_1}, \mathbf{H}''_{1,1}, \ldots, \mathbf{H}''_{i,k}, \ldots, \mathbf{H}''_{n,\ell_n}$ where $\mathbf{H}''_{1,0} = \mathbf{G}_3$ and $\mathbf{H}''_{n,\ell_n} = \mathbf{G}_4$. The game $\mathbf{H}''_{\rho,\delta}$ is defined as follows:

Game $\mathbf{H}''_{\rho,\delta}$. This game $\mathbf{H}''_{\rho,\delta}$ is almost identical to the game $\mathbf{G}_3$ except the generation of components $\{D_{i,k}\}$ in the challenge ciphertexts.

- **Case** $(i < \rho)$ or $(i = \rho \wedge k \leq \delta)$: If $x^*_{\mu,i,k} \in E^*_i$, then the component $D_{i,k}$ is generated as normal. Otherwise ($x^*_{\mu,i,k} \notin E^*_i$), the component $D_{i,k}$ is generated as the encryption of a random value.
- **Case** $(i = \rho \wedge k > \delta)$ or $(i > \rho)$: The component $D_{i,k}$ is generated as normal.

Suppose there exists an adversary $\mathcal{A}$ that distinguishes between $\mathbf{H}''_{\rho,\delta-1}$ and $\mathbf{H}''_{\rho,\delta}$ with a non-negligible advantage. Without loss of generality, we assume that $x^*_{\mu,\rho,\delta} \notin E^*_\rho$ since $\mathbf{H}''_{\rho,\delta-1}$ and $\mathbf{H}''_{\rho,\delta}$ are equal if $x^*_{\mu,\rho,\delta} \in E^*_\rho$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ submits challenge sets of item sets $\{X^*_{0,1}, \ldots, X^*_{0,n}\}$, $\{X^*_{1,1}, \ldots, X^*_{1,n}\}$, a challenge time period T^*, and a set $Q = \{(i,j)\}$ of function key queries. $\mathcal{B}$ then flips a random coin $\mu \in \{0,1\}$ internally and derives a set $\{E^*_1, \ldots, E^*_n\}$ of common sets by calling $CIQ(\{X^*_{\mu,k}\}, Q)$.

Setup: $\mathcal{B}$ first chooses random exponents $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \in \mathbb{Z}_p$. Next, it sets $PP = ((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e), g, \hat{g}, H_1, H_2)$. It prepares a hash table H-list for the H_1 hash function as follows:

1. For each $i \in [n]$ and $k \in [\ell_i]$, it selects a random exponent $f_{i,k} \in \mathbb{Z}_p$ and adds $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ to the H-list.

Challenge: $\mathcal{B}$ creates challenge ciphertexts $CT_{1,T^*}, \ldots, CT_{n,T^*}$ as follows:

1. For each i and k, it generates ciphertext elements $C_{i,k}$ and $TK_{i,k}$ depending on the following cases:
   - Case $x^*_{\mu,i,k} \in E_i$: It retrieves $(T^* \| x^*_{\mu,i,k}, f_{i,k}, g^{f_{i,k}})$ from the H-list, and creates $C_{i,k} = g^{f_{i,k}\alpha_i}$ and $TK_{i,k} = e(g^{f_{i,k}}, \hat{g})^{\beta_i}$.
   - Case $x^*_{\mu,i,k} \notin E_i$: It selects random $C_{i,k} \in \mathbb{G}$ and random $TK_{i,k} \in \mathbb{G}_T$.

   Next, it also generates a ciphertext element $D_{i,k}$ depending on the following cases:

   - Case $(i < \rho)$ or $(i = \rho \wedge k < \delta)$: If $x^*_{\mu,i,k} \in E^*_i$, it creates $D_{i,k}$ by running **SKE.Encrypt**$(T^* \| x^*_{\mu,i,k}, TK_{i,k})$. Otherwise ($x^*_{\mu,i,k} \notin E^*_i$), it selects a random $y \in \mathcal{D}$ and creates $D_{i,k}$ by running **SKE.Encrypt** $(T^* \| y, TK_{i,k})$.
   - Case $(i = \rho \wedge k = \delta)$: It selects a random $y \in \mathcal{D}$ and submits challenge message $x^*_{\mu,\rho,\delta}$ and y to the encryption oracle of SKE. Next, it receives a challenge ciphertext CT^*_{SKE} from SKE and sets $D_{\rho,\delta} = CT^*_{SKE}$. Recall that we assumed $x^*_{\mu,\rho,\delta} \notin E^*_\rho$.
   - Case $(i = \rho \wedge k > \delta)$ or $(i > \rho)$: It creates $D_{i,k}$ by running **SKE.Encrypt**$(T^* \| x^*_{\mu,i,k}, TK_{i,k})$.

2. It chooses a random permutation π_i and sets a challenge ciphertext $CT_{i,T^*} = \{(C_{i,\pi_i(k)}, D_{i,\pi_i(k)})\}_{k=1}^{\ell_i}$ for each client i.

Query: $\mathcal{A}$ adaptively requests hash, function key, and ciphertext queries. $\mathcal{B}$ handles these queries as follows:

- If this is a hash query for a time period T and an item x, then $\mathcal{B}$ proceeds as follows: If $T \| x$ exists in the H-list, then it retrieves $(T \| x, -, h)$ from H-list and gives h to $\mathcal{A}$. Otherwise, it selects a random exponent $f \in \mathbb{Z}_p$ and adds $(T \| x, f, g^f)$ to the H-list, and then it gives the hash value g^f to $\mathcal{A}$.

- If this is a function key query for indexes i, j, then $\mathcal{B}$ simply generates a function key $SK_{i,j}$ by using $\alpha_i, \alpha_j, \beta_i$.
- If this is a ciphertext query for a client index i, a set X_i, and a time period $T \neq T^*$, then $\mathcal{B}$ simply generates a ciphertext CT by using α_i, β_i.

Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, it outputs 1. Otherwise, it outputs 0. □

Now, we analyze the static security with corrupted clients of the FE-SI scheme. In the definition of the static security model, two indexes of a function key must be associated with corrupted clients, or two indexes of the function key must be associated with non-corrupted clients. Therefore, we can process this proof by selecting encryption keys for corrupted clients and using the static security proof of the FE-SI scheme analyzed earlier for non-corrupted clients.

Theorem 3.6. *The above FE-SI scheme is static-IND secure with corruptions in the random oracle model if the FE-SI scheme is static-IND secure with no corruptions.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the static-IND security with corruptions. By using $\mathcal{A}$, a simulator $\mathcal{B}$ try to break the static-IND security with no corruptions played by a challenger $\mathcal{C}$. The simulator $\mathcal{B}$ is described as follows:

Init: $\mathcal{A}$ submits the set of corrupted client indexes $\bar{I} \neq \emptyset$. Let $I = \{1, \ldots, n\} \setminus \bar{I}$ be the set of uncorrupted client indexes where $|I| = n'$. $\mathcal{A}$ also submits two challenge sets $\{X^*_{0,k}\}_{k \in I}, \{X^*_{1,k}\}_{k \in I}$, a challenge time period T^*, and a set $Q = \{(i, j)\}$ of function key queries.

1. It first define a one-to-one mapping ϕ from I to $I' = \{1, \ldots, n'\}$ such that $\phi(i) < \phi(j)$ if $i < j$ for any $i, j \in I$. It also define ϕ^{-1} as the inverse mapping of ϕ.
2. Next, it derives a new set $Q' = \{(\phi(i), \phi(j)) : (i, j) \in Q\}$ from the set Q.
3. $\mathcal{B}$ submits two challenge sets $\{X^*_{0,\phi(k)}\}_{\phi(k) \in I'}, \{X^*_{1,\phi(k)}\}_{\phi(k) \in I'}$, the challenge time period T^*, and the set Q' to $\mathcal{C}$. Note that $\mathcal{C}$ plays the static-IND security game with no corruptions for the set I'.

Setup: $\mathcal{B}$ receives PP from $\mathcal{C}$. It chooses random exponents $\{\alpha_i, \beta_i\}_{i \in \bar{I}}$. Next, it gives $\{EK_i = (\alpha_i, \beta_i)\}_{i \in \bar{I}}$ and PP to $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives challenge ciphertexts $\{CT_{i',T^*}\}_{i' \in I'}$ from $\mathcal{C}$ and gives $\{CT_{\phi^{-1}(i'),T^*}\}_{\phi^{-1}(i') \in I}$ to $\mathcal{A}$.

Query: $\mathcal{A}$ requests hash, function key, and ciphertext queries. $\mathcal{B}$ relays these queries to $\mathcal{C}$ and gives the response of $\mathcal{C}$ to $\mathcal{A}$ by using the mappings ϕ and ϕ^{-1} to change the indexes of sets I' and I.

Guess: $\mathcal{A}$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

3.6 Discussions

Efficiency Analysis. We analyze the efficiency of our FE-SI scheme. First, the encryption algorithm requires 2ℓ exponentiation operations and 2ℓ symmetric key encryption operations since it computes $C_{i,k}, TK_{i,k}$, and $D_{i,k}$ for each item of the set where ℓ is the size of the set. The size of the ciphertext is proportional to the size of the set. The key generation algorithm is efficient because it only requires a PRF computation and a single exponentiation operation, and the function key is composed of one group element. The slowest part of the FE-SI scheme is the decryption algorithm. Since the decryption algorithm processes two possible combination of ciphertext elements of two clients, it requires $O(\ell^2)$ pairing operations and $O(\ell^2)$ symmetric-key decryptions. Therefore, the decryption algorithm has $O(\ell^2)$ time complexity when ℓ is the size of the set.

Hiding the Set Sizes. In our FE-SI scheme, the ciphertext exposes the set size information because the size of a ciphertext is proportional to the size of the set. A simple way to hide the size of a set is to include additional dummy items in the ciphertext. That is, a dummy element $\tilde{C}_{i,k}$ is generated by computing $H(r)^{\alpha_i}$ with a random string r such that $\mathrm{prefix}(r) \neq T$, and a dummy element $\tilde{D}_{i,k}$ is generated by running a symmetric-key encryption on a message r. In this case, the probability that the two clients select the same random string r is very low, and even if the two clients select the same r, the time string T is not decoded from the decryption of the dummy $\tilde{D}_{i,k}$. Thus the intersection of ciphertexts with dummy elements works correctly.

Encryption with Associated Data. The FE-SI scheme encrypts only the set information during encryption. One natural way to extend the FE-SI scheme is to encrypt a set with additional associated data. This associated data can be easily encrypted by using symmetric-key encryption. In this case, the entity with a function key can decrypt not only the set intersection of two clients but also associated data encrypted by individual clients. Since the function key of our FE-SI scheme only provides $\hat{g}^{\beta_i/(\alpha_i+\alpha_j)}$ for client indexes i, j, only the associated data of the client index i is decrypted. If only the set is encrypted, this is not a problem, but if associated data is included, it is necessary to decrypt the ciphertext elements of the client j as well as the client i. Therefore, for this purpose, the key generation algorithm must provide two group elements, $\hat{g}^{\beta_i/(\alpha_i+\alpha_j)}$ and $\hat{g}^{\beta_j/(\alpha_i+\alpha_j)}$ as function keys.

Set Intersection Cardinality. An interesting variant of the functional encryption for set intersection is functional encryption for set intersection cardinality (FE-SIC) that reveals the cardinality of the set intersection instead of revealing the set intersection of two sets. The simplest way to devise an FE-SIC scheme is to modify our FE-SI scheme to encrypt the string T instead of the concatenated string $T\|x$ when generating the ciphertext element $D_{i,k}$. In this case, if the correct string T is derived during decryption, the same item exists, so the number of intersections can be calculated by counting these cases. However, this method has the disadvantage of preserving the decryption complexity $O(\ell^2)$ of the FE-SI scheme and requiring to create a new ciphertext.

Multi-Party Set Intersection. The FE-SI scheme we devised calculates the set intersection of two clients. It is possible to modify the FE-SI scheme to calculate the set intersection of three clients. The basic idea is for the key generation algorithm to receive the indexes of three clients (i, j, k) as an input and to create a function key $SK_{i,j,k} = \hat{g}^{\beta_i/(\alpha_i+\alpha_j+\alpha_k)}$. In this case, if the ciphertexts generated by the clients of indexes i, j, and k are on the same item x, then the decryption algorithm first derives $H(T\|x)^{\alpha_i} \cdot H(T\|x)^{\alpha_j} \cdot H(T\|x)^{\alpha_k} = H(T\|x)^{\alpha_i+\alpha_j+\alpha_k}$. If the pairing operation is performed with the function key, the temporary key $e(H(T\|x), \hat{g})^{\beta_i}$ can be derived for symmetric-key decryption. The decryption algorithm requires $O(\ell^3)$ pairing operations since all possible combination of ciphertext elements should be considered. The static security of this scheme with no corruptions can be easily proven by extending the constraints of the security model and using modified assumptions that include additional group elements for key queries. If this method is extended, it is possible to process the set intersection of n clients, but it is inefficient because the decryption algorithm has $O(\ell^n)$ time complexity.

4 FE for Set Intersection with Time-Constrained Keys

In this section, we define the syntax and the security model of FE-SI that supports time-constrained keys (FE-SI-TCK). In addition, we propose an FE-SI-TCK scheme by modifying the previous FE-SI scheme and prove the security of the scheme.

4.1 Definition

The syntax of FE-SI-TCK is almost similar to that of FE-SI. The key difference in the syntax of FE-SI-TCK is that a function key is only valid at time T because the function key is associated with client indexes i, j and additional time T. For this reason, the decryption algorithm of FE-SI-TCK correctly proceeds the decryption process only when the time T of the two client's ciphertexts and the time T of the function key are the same. That is, the function key of the FE-SI-TCK scheme does not always compute the set intersection of the two client's ciphertexts for any time, but only computes the set intersection of the two client's ciphertexts corresponding to the limited time T. A more detailed syntax of FE-SI-TCK is defined as follows.

Definition 4.1 (FE-SI with Time-Constrained Keys)**.** An FE-SI with time-constrained keys (FE-SI-TCK) scheme for $\mathcal{D}$ and $\mathcal{T}$ consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

Setup$(1^\lambda, n)$. The setup algorithm takes as input a security parameter λ and the number of clients n. It outputs a master key MK, encryption keys $\{EK_i\}$ for clients, and public parameters PP.

GenKey(i, j, T, MK, PP). The key generation algorithm takes as input two client indexes i, j such that $i < j$, a time period T, the master key MK, and public parameters PP. It outputs a time-constrained function key $SK_{i,j,T}$.

Encrypt(X_i, T, EK_i, PP). The encryption algorithm takes as input a set $X_i = (x_{i,1}, \ldots, x_{i,\ell_i})$ where $x_{i,j} \in \mathcal{D}$, a time period $T \in \mathcal{T}$, an encryption key EK_i, and the public parameters PP. It outputs a ciphertext $CT_{i,T}$.

Decrypt$(CT_{i,T}, CT_{j,T}, SK_{i,j,T}, PP)$. The decryption algorithm takes as input two ciphertexts $CT_{i,T}$ and $CT_{j,T}$, a function key $SK_{i,j,T}$, and the public parameters PP. It outputs a set $X_i \cap X_j$ where X_i and X_j are associated with $CT_{i,T}$ and $CT_{j,T}$ respectively.

The correctness property of FE-SI-TCK is defined as follows: For all $MK, \{EK_i\}, PP \leftarrow$ **Setup**$(1^\lambda, n)$, any $SK_{i,j,T} \leftarrow$ **GenKey**(i, j, T, MK, PP) for any i, j, T such that $i < j$, $CT_{i,T} \leftarrow$ **Encrypt**(X_i, T, EK_i, PP), and $CT_{j,T} \leftarrow$ **Encrypt**(X_j, T, EK_j, PP), it is required that

- **Decrypt**$(CT_{i,T}, CT_{j,T}, SK_{i,j,T}, PP) = X_i \cap X_j$ except with negligible probability.

The static security model of FE-SI-TCK is almost similar to the previous defined static security model of FE-SI. An important difference is that the attacker of FE-SI-TCK initially submits a set Q_{T^*} of function key queries related to a challenge time T^* instead of submitting all function key queries. Afterwards, when the attacker queries a function key for time T, the function key query must be specified in Q_{T^*} if $T = T^*$, but the function key query can be any query if $T \neq T^*$. A more detailed static security model of FE-SI-TCK is defined as follows.

Definition 4.2 (Static-IND Security)**.** The static-IND security of FE-SI-TCK with corruptions is defined in the following experiment $\mathbf{EXP}^{\textit{ST-IND}}_{\textit{FE-SI-TCK},\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. **Init**: $\mathcal{A}$ initially submits an index set $\bar{I} \subset [n]$ of corrupted clients. Let $I = \{1, \ldots, n\} \setminus \bar{I}$ be the index set of uncorrupted clients. $\mathcal{A}$ also submits two challenge sets of item sets $\{X^*_{0,k}\}_{k \in I}$, $\{X^*_{1,k}\}_{k \in I}$, a challenge time period T^*, and a set $Q_{T^*} = \{(i, j)\}$ of function key queries on the time period T^* with the restriction that $i, j \in I$ for each $(i, j) \in Q$ and $CIQ(\{X^*_{0,k}\}_{k \in I}, Q_{T^*}) = CIQ(\{X^*_{1,k}\}_{k \in I}, Q_{T^*})$.

2. **Setup**: $\mathcal{C}$ generates a master key MK, encryption keys $\{EK_i\}_{i=1}^n$ and public parameters PP by running **Setup**$(1^\lambda, n)$. It keeps MK and $\{EK_i\}_{i\in I}$ to itself and gives $\{EK_i\}_{i\in \bar{I}}$ and PP to $\mathcal{A}$.

3. **Challenge**: $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and obtains a ciphertext CT_{i,T^*} by running **Encrypt**$(X^*_{\mu,i}, T^*, EK_i, PP)$ for each $i \in I$. $\mathcal{C}$ gives the challenge ciphertexts $\{CT_{i,T^*}\}_{i\in I}$ to $\mathcal{A}$

4. **Query**: $\mathcal{A}$ requests function keys and ciphertexts. $\mathcal{C}$ handles these queries as follows:

   - If this is a function key query for indexes $i, j \in I$ and a time period T with the restriction that $(i,j) \in Q_{T^*}$ if $T = T^*$, then $\mathcal{C}$ gives a function key $SK_{i,j,T}$ to $\mathcal{A}$ by running **GenKey**(i,j,T,MK,PP).
   - If this is a ciphertext query for a client index k, a set X_k, and a time period T with the restriction that $k \in I$ and $T \neq T^*$, then $\mathcal{C}$ gives $CT_{k,T}$ to $\mathcal{A}$ by running **Encrypt**(X_k, T, EK_k, PP).

5. **Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$ of μ. $\mathcal{C}$ outputs 1 if $\mu = \mu'$ or 0 otherwise.

An FE-SI-TCK scheme is static-IND secure with corruptions if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as $\mathbf{Adv}^{ST\text{-}IND}_{\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{EXP}^{ST\text{-}IND}_{FE\text{-}SI\text{-}TCK,\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}\right|$ is negligible in the security parameter λ.

4.2 Construction

The idea of devising a function key limited to time T is to derive an independent encryption key for each time T from an original client encryption key by slightly modifying our FE-SI scheme. In other words, the client encryption key of the FE-SI-TCK scheme is a PRF key z and two exponents $(\alpha_T = \mathrm{PRF}(z, 1\|T), \beta_T = \mathrm{PRF}(z, 2\|T))$ are derived for a specific time period T, whereas the client encryption key of the FE-SI scheme is just (α, β). The encryption and decryption algorithms are almost the same as those of the FE-SI scheme. An FE-SI-TCK scheme that supports time-constrained function keys is described as follows.

Setup$(1^\lambda, n)$. Let n be the maximum number of clients.

1. It first generates a bilinear group $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ of prime order p with two random generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. It chooses two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \mathbb{G}_T \to \{0,1\}^\lambda$.
2. Next, it selects a random PRF key $z \in \{0,1\}^\lambda$ and computes $z_i = PRF(z, 0\|i)$ for each index $i \in [n]$.
3. Finally, it outputs a master key $MK = z$, encryption keys $\{EK_i = z_i\}_{i=1}^n$ for clients, and public parameters $PP = \big((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e), g, \hat{g}, H_1, H_2\big)$.

GenKey(i, j, T, MK, PP). Let i, j be indexes such that $i < j$ and T be a time period.

1. It first derives $z_i = PRF(z, 0\|i)$ and $z_j = PRF(z, 0\|j)$ from $MK = z$. It also calculates $\alpha_{i,T} = PRF(z_i, 1\|T)$, $\alpha_{j,T} = PRF(z_j, 1\|T)$, and $\beta_{i,T} = PRF(z_i, 2\|T)$.
2. Finally, it outputs a time-constrained function key $SK_{i,j,T} = \hat{g}^{\beta_{i,T}/(\alpha_{i,T}+\alpha_{j,T})}$ by implicitly including i, j, T.

Encrypt(X_i, T, EK_i, PP). Let $X_i = \{x_{i,1}, \ldots, x_{i,\ell_i}\}$ be a set of items where $|X_i| = \ell_i$ and $EK_i = z_i$.

1. It calculates $\alpha_{i,T} = PRF(z_i, 1\|T)$ and $\beta_{i,T} = PRF(z_i, 2\|T)$.
2. For each index $k \in [\ell_i]$, it proceed as follows: It computes a component $C_{i,k} = H_1(x_{i,k})^{\alpha_{i,T}}$ and then derives a temporal key $TK_{i,k} = e(H_1(x_{i,k}), \hat{g})^{\beta_{i,T}}$. Next, it obtains an encrypted data $D_{i,k}$ by running **SKE.Encrypt**$(T\|x_{i,k}, H_2(TK_{i,k}))$.

3. Finally, it chooses a random permutation π and outputs a ciphertext $CT_{i,T} = \left\{(C_{i,\pi(k)}, D_{i,\pi(k)})\right\}_{k=1}^{\ell_i}$ by implicitly including i, T.

Decrypt$(CT_{i,T}, CT_{j,T}, SK_{i,j,T}, PP)$. Let $CT_{i,T} = \{(C_{i,k}, D_{i,k})\}_{k=1}^{\ell_i}$ and $CT_{j,T} = \{(C_{j,k}, D_{j,k})\}_{k=1}^{\ell_j}$ be ciphertexts such that $i < j$ with the same time T. It first initializes a set $Y = \emptyset$.

1. For each index $k_i \in [\ell_i]$ and $k_j \in [\ell_j]$, it proceeds as follows: It computes $TK_{i,k_i} = e(C_{i,k_i} \cdot C_{j,k_j}, SK_{i,j,T})$ and obtains a message $A \| x$ by running **SKE.Decrypt**$(D_{i,k_i}, H_2(TK_{i,k_i}))$. It adds an item x into Y if $A = T$.
2. Finally, it outputs the set Y.

4.3 Correctness

The FE-SI-TCK scheme is almost the same as the previous FE-SI scheme except that the encryption key for a specific time period T is derived by using the PRF. Therefore, the correctness of the FE-SI-TCK scheme is easily guaranteed by the correctness of the PRF and the correctness of the FE-SI scheme.

4.4 Security Analysis

In order to prove the static security of the FE-SI-TCK scheme, we also analyze by dividing into two cases, the case where there is no corrupted client and the case where there are corrupted clients. Similar to the proof of the FE-SI scheme, we can also prove the static security of the scheme with corrupted clients by using the static security proof of the scheme with non-corrupted clients.

We can show that the FE-SI-TCK scheme without corrupted clients is static secure by using the static security of the FE-SI scheme without corrupted clients. The basic idea of this proof is that ciphertext and function key queries for $T = T^*$ are all handled by the challenger of the FE-SI scheme, and ciphertext and function key queries for $T \neq T^*$ are handled by a simulator itself with randomly selected encryption keys. The security theorem of the FE-SI-TCK scheme is described as follows.

Theorem 4.1. *The above FE-SI-TCK scheme is static-IND secure with no corruptions in the random oracle model if the PRF scheme is secure and the FE-SI scheme is static-IND secure with no corruptions.*

Proof. To argue that the adversary cannot win this game, we define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1$. The game $\mathbf{G}_i$ is defined as follows:

Game $\mathbf{G}_0$. The first game $\mathbf{G}_0$ is the original security game defined in Definition 4.2.

Game $\mathbf{G}_1$. In this game $\mathbf{G}_1$, the PRFs which are used to generate encryption keys are changed to be truly random functions.

Let $S_{\mathcal{A}}^{\mathbf{G}_i}$ be the event that an adversary wins in a game $\mathbf{G}_i$. From the following lemmas 4.2 and 4.3, we obtain the following result

$$\mathbf{Adv}_{FE\text{-}SI\text{-}TCK,\mathcal{A}}^{ST\text{-}IND}(\lambda) \leq \left|\Pr[S_{\mathcal{A}}^{\mathbf{G}_0}] - \Pr[S_{\mathcal{A}}^{\mathbf{G}_1}]\right| + \Pr[S_{\mathcal{A}}^{\mathbf{G}_1}] \leq (n+1)\mathbf{Adv}_{\mathcal{B}}^{PRF}(\lambda) + \mathbf{Adv}_{FE\text{-}SI,\mathcal{B}}^{ST\text{-}IND}(\lambda)$$

where n is the number of clients, ℓ is the size of the challenge attribute. This completes our proof. □

Lemma 4.2. *If the PRF is secure, then no polynomial-time adversary can distinguish between $\mathbf{G}_0$ and $\mathbf{G}_1$ with a non-negligible advantage.*

Proof. The proof of this lemma is relatively easy from the security of PRF. That is, we simply change a PRF to a truly random function one by one by defining additional hybrid games. Note that there are at most $n+1$ PRFs are used in the FE-SI-TCK scheme. We omit the details of this proof. □

Lemma 4.3. *If the FE-SI scheme is static-IND secure, then no polynomial-time adversary can win* $\mathbf{G}_1$ *with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that wins the game $\mathbf{G}_1$ with a non-negligible advantage. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ submits challenge sets of item sets $\{X^*_{0,1},\ldots,X^*_{0,n}\}$, $\{X^*_{1,1},\ldots,X^*_{1,n}\}$, a challenge time period T^*, and a set Q_{T^*} of function key queries. $\mathcal{B}$ also submits $\{X^*_{0,i}\}$, $\{X^*_{1,i}\}$, T^*, and Q_{T^*} to the FE-SI scheme.

Setup: $\mathcal{B}$ receives $\tilde{PP}$ from the FE-SI scheme and sets $PP=\tilde{PP}$. $\mathcal{B}$ prepares an EK-list as the empty set that stores a tuple $(i,T,\alpha_{i,T},\beta_{i,T})$.

Challenge: $\mathcal{B}$ receives $\tilde{CT}_{1,T^*},\ldots,\tilde{CT}_{n,T^*}$ of the FE-SI scheme and sets challenge ciphertexts $CT_{i,T^*}=\tilde{CT}_{i,T^*}$ for each i. It gives the challenge ciphertexts to $\mathcal{A}$.

Query: $\mathcal{A}$ adaptively requests hash, function key, and ciphertext queries. $\mathcal{B}$ handles these queries as follows:

- If this is a hash query for an item x, then $\mathcal{B}$ proceeds as follows: It requests a hash query to the FE-SI scheme on the time period T^* and the item x and receives a hash value $\tilde{h}$. It sets $h=\tilde{h}$ and gives h to $\mathcal{A}$. Note that it implicitly sets $H_1(x)=\tilde{H}_1(T^*\|x)$ where $\tilde{H}_1$ is the hash function of the FE-SI scheme.
- If this is a time-constrained function key query for indexes i,j and a time period T, then $\mathcal{B}$ generates a function key as follows:
  - Case $T=T^*$: It requests a ciphertext query to the FE-SI scheme on input i,j and receives a function key $\tilde{SK}_{i,j}$ since $(i,j)\in Q_{T^*}$. It creates a time-constrained function key $SK_{i,j,T}=\tilde{SK}_{i,j}$.
  - Case $T\neq T^*$:
    1. If a tuple $(i,T,\alpha_{i,T},\beta_{i,T})$ already exists in the EK-list, then it retrieves $(i,T,\alpha_{i,T},\beta_{i,T})$ from the EK-list. Otherwise, it selects random $\alpha_{i,T},\beta_{i,T}\in\mathbb{Z}_p$ and adds $(i,T,\alpha_{i,T},\beta_{i,T})$ to the EK-list.
    2. If a tuple $(j,T,\alpha_{j,T},\beta_{j,T})$ already exists in the EK-list, then it retrieves $(j,T,\alpha_{j,T},\beta_{j,T})$ from the EK-list. Otherwise, it selects random $\alpha_{j,T},\beta_{j,T}\in\mathbb{Z}_p$ and adds $(j,T,\alpha_{j,T},\beta_{j,T})$ to the EK-list.
    3. Next, it creates a time-constrained function key $SK_{i,j,T}=\hat{g}^{\beta_{i,T}/(\alpha_{i,T}+\alpha_{j,T})}$.
- If this is a ciphertext query for a client index i, a set $X_i=\{x_{i,1},\ldots,x_{i,\ell_i}\}$, and a time period $T\neq T^*$, then $\mathcal{B}$ generates a ciphertext as follows:
  1. If a tuple $(i,T,\alpha_{i,T},\beta_{i,T})$ already exists in the EK-list, then it retrieves $(i,T,\alpha_{i,T},\beta_{i,T})$ from the EK-list. Otherwise, it selects random $\alpha_{i,T},\beta_{i,T}\in\mathbb{Z}_p$ and adds $(i,T,\alpha_{i,T},\beta_{i,T})$ to the EK-list.
  2. For each index $k\in[\ell_i]$, it obtains h from hash query on input $x_{i,k}$ and creates $C_{i,k}=h^{\alpha_{i,T}}$, $TK_{i,k}=e(h,\hat{g})^{\beta_{i,T}}$, and $D_{i,k}$ by running $\mathbf{SKE.Encrypt}(T\|x_{i,k},H_2(TK_{i,k}))$.
  3. It chooses a random permutation π and generates a ciphertext $CT_{i,T}=\{(C_{i,\pi(k)},D_{i,\pi(k)})\}_{k=1}^{\ell_i}$.

Guess: $\mathcal{A}$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

Theorem 4.4. *The above FE-SI-TCK scheme is static-IND secure with corruptions in the random oracle model if the FE-SI-TCK scheme is static-IND secure with no corruptions.*

The proof of Theorem 4.4 is almost identical to that of Theorem 3.6.

4.5 Discussions

Time Range Function Key. In our FE-SI-TCK scheme, a function key is only valid for a specified time period T. If a third party performing the set intersection operation needs a function key that is valid for time range periods from T_L to T_R, the trusted center generates a time range function key which consists of individual function keys for each time period $T \in [T_L, T_R]$. In this case, the size of the time range function key increases in proportion to the size of the time range. We can reduce the size of the time range function key by using a binary tree. That is, if the leaf nodes in the binary tree are related to time periods, a ciphertext is related to the path of the binary tree, and a time range function key is related to the minimum set of internal nodes which include all leaf nodes corresponding to the time range. In this case, the ciphertext is composed of $2\ell \log(T_{max})$ group elements, and the time range function key can also be composed of $2\log(T_{max})$ group elements where T_{max} is the maximum time period of the binary tree.

Forward Secrecy. A forward-secure encryption scheme does not expose information on ciphertexts generated in the past time periods if a long-term key is exposed to an attacker [13]. In our FE-SI-TCK scheme, if the encryption key of an corrupted client, which is the client's long-term key, is exposed, the attacker can obtain information on the past ciphertexts of another uncorrupted client by using the function key of the past time and the encryption key of the corrupted client. Note that the security model of FE-SI-TCK is not considered forward secrecy because corrupt clients are fixed in advance. We can modify our FE-SI-TCK scheme to provide forward secrecy. First, each client is initially given an encryption key (α_0, β_0), and the encryption key evolves over time through $\alpha_T = \mathrm{PRG}(\alpha_{T-1})$ and $\beta_T = \mathrm{PRG}(\beta_{T-1})$. The encryption algorithm uses the evolved encryption key (α_T, β_T) to generate a ciphertext as the same as that of our FE-SI-TCK scheme. If an attacker obtains an encryption key (α_T, β_T) on time T, the attacker can generate a future ciphertext on time $T'' \geq T$, but he cannot generate a past ciphertext on time $T' < T$ due to the one-wayness of pseudo-random generators.

5 Our Assumptions in Generic Group Models

In this section, we prove that our assumptions we introduced earlier hold in the generic group model [33]. To do this, we use the master theorem in [7, 20] to analyze our assumptions in asymmetric bilinear groups.

5.1 Generic Group Models

The generic group model is an abstract model for analyzing generic algorithms that operate independently of the representation of group elements. For this reason, just because an assumption is analyzed to be secure in the generic group model does not guarantee that the assumption is secure in the real environment. The reason is that there may be a non-generic algorithm that breaks the assumption.

A generic group model is defined as a game between a challenger algorithm $\mathcal{B}$ and an attacker algorithm $\mathcal{A}$. In this case, $\mathcal{A}$ does not directly access the actual representation of group elements, but instead uses unique handles to these elements to access these group elements. That is, $\mathcal{A}$ can query group multiplication, division, exponentiation, and pairing operations by using handles of group elements, and can compare the equality of group elements by directly comparing the handles. Initially, $\mathcal{A}$ is given the handles of group

elements with the specified distribution. The processing of the handling the query of $\mathcal{A}$ is described as follows:

- **Multiplication**: $\mathcal{A}$ queries the multiplication operation by submitting two handles h_1 and h_2 associated with group elements $u_1 = g^{x_1}$ and $u_2 = g^{x_2}$. To answer this query, $\mathcal{B}$ computes $u_3 = u_1 u_2 = g^{x_1+x_2}$ and returns a new handle h_3 if the element u_3 has not already been assigned to a handle. If u_3 already has a handle, it returns the existing handle. The multiplication operation is processed in a similar way for each group $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ respectively.

- **Division**: $\mathcal{A}$ queries the division operation by submitting two handles h_1 and h_2 associated with group elements $u_1 = g^{x_1}$ and $u_2 = g^{x_2}$. To answer this query, $\mathcal{B}$ computes the element $u_3 = u_1/u_2 = g^{x_1-x_2}$ and returns a new handle h_3 if u_3 has not already been assigned a handle. If u_3 already has a handle, it returns the existing handle. The division operation is processed in a similar way for each group $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ respectively.

- **Exponentiation**: $\mathcal{A}$ queries the exponentiation operation by submitting a handle h_1 associated with a group element $u_1 = g^{x_1}$ and an integer γ. To answer this query, $\mathcal{B}$ computes the element $u_2 = u_1^{\gamma} = g^{x_1\gamma}$ and returns a new handle h_2 if u_2 has not already been assigned to a handle. If u_2 already has a handle, it returns the existing handle. The exponentiation operation is processed in a similar way for each group $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ respectively.

- **Pairing**: $\mathcal{A}$ queries the pairing operation by submitting two handles h_1 and h_2 associated with group elements $u_1 = g^x$ and $\hat{u}_2 = \hat{g}^y$. To answer this query, $\mathcal{B}$ computes the element $u_T = e(u_1, \hat{u}_2) = e(g, \hat{g})^{xy}$ and returns a new handle h_3 if u_T is not already assigned to a handle in the group $\mathbb{G}_T$. If u_T already has a handle, it returns the existing handle.

Let $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ be asymmetric bilinear groups of prime order p with the bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. A group element $u \in \mathbb{G}$ can be represented as a multi-variate polynomial, which indicates the exponent of u relative to some fixed generator g. We can also represent group elements in $\hat{\mathbb{G}}$ and $\mathbb{G}_T$ as similar way. For instance, the general Diffie-Hellman tuple is represented as the tuple $(1, X, Y, XY)$ where X and Y are random variables.

The generalized dependence and independence of variables is defined by Freeman [20] as follows:

Definition 5.1 ([20], Definition D.1)**.** Let $P = (p_1, \ldots, p_u)$, $R = (r_1, \ldots, r_w)$, $T = (t_1, \ldots, t_v)$, $S = (s_1, \ldots, s_t)$ be tuples of multi-variate polynomials in $\mathbb{F}_p[X_1, \ldots, X_n]$ where X_i is a random variable. Let f be a multi-variate polynomial in $\mathbb{F}_p[X_1, \ldots, X_n]$. We say that $f \cdot S$ is *dependent on* (P, R, T) if there exist integers $\{\alpha_{i,j}\}, \{\beta_k\}, \{\gamma_\ell\}$ such that

$$\sum_{i=1}^{u} \sum_{j=1}^{w} \alpha_{i,j} \cdot p_i r_j + \sum_{k=1}^{v} \beta_k \cdot t_k + \sum_{\ell=1}^{t} \gamma_\ell \cdot Y s_\ell$$

is nonzero in $\mathbb{F}_p[X_1, \ldots, X_n, Y]$ but becomes zero when we set $Y = f$. We say that $f \cdot S$ is *independent* of (P, R, T) if $f \cdot S$ is not dependent on (P, R, T). We say that f is *independent* of (P, R, T) if $f \cdot \{1\}$ is not dependent on (P, R, T).

In this definition, the multi-variate polynomials p_i, r_j, t_k represent the exponents of group elements in $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ respectively, and the polynomial f represents the exponent of the challenge element in complexity assumptions. Additionally, the polynomials s_ℓ represent the exponents of group elements in which the challenge element can be paired.

By extending the (P, R, T, f)-DDH problem of Boneh et al. [7], Freeman defined the (P, R, T, f)-DDH problem in $\mathbb{G}$ and $\mathbb{G}_T$ as follows:

Definition 5.2 ([20], Definition D.2)**.** Let $(p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e)$ be a bilinear group randomly generated by $\mathcal{G}(1^\lambda)$. Let $g,\hat{g}$ be random generators of $\mathbb{G},\hat{\mathbb{G}}$ respectively. Let P,R,T,f be as in Definition 5.1. We select $\vec{x} \xleftarrow{R} \mathbb{F}_p^n$ and define the following distribution:

$$D = \big((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),\ g^{p_1(\vec{x})},\ldots,g^{p_u(\vec{x})},\ \hat{g}^{r_1(\vec{x})},\ldots,\hat{g}^{r_w(\vec{x})},\\ e(g,\hat{g})^{t_1(\vec{x})},\ldots,e(g,\hat{g})^{t_v(\vec{x})}\big),\ Z_0 \leftarrow g^{f(\vec{x})},\ Z_1 \xleftarrow{R} \mathbb{G}$$

We define the advantage of an algorithm $\mathcal{A}$ that outputs $b \in \{0,1\}$ in solving the (P,R,T,f)-decision Diffie-Hellman problem in $\mathbb{G}$ to be

$$\mathbf{Adv}_{\mathcal{A}}^{(P,R,T,f)\text{-}DDH}(\lambda) = |\Pr[\mathcal{A}(D,Z_0)=1] - \Pr[\mathcal{A}(D,Z_1)=1]|$$

We define the analogous problem in $\mathbb{G}_T$ by taking $Z_0 \leftarrow e(g,\hat{g})^{f(\vec{x})}, Z_1 \xleftarrow{R} \mathbb{G}_T$.

The master theorem of Boneh et al. [7] gives the complexity lower bound of the (P,R,T,f)-DDH problem in $\mathbb{G}_T$, but the same argument also works for the (P,R,T,f)-DDH problem in $\mathbb{G}$ as indicated by Freeman [20] using the generalized definition of independence in Definition 5.1.

Theorem 5.1 ([7, 20])**.** *Let $P=(p_1,\ldots,p_u)$, $R=(r_1,\ldots,r_w)$, $T=(t_1,\ldots,t_v)$ be tuples of polynomials in $\mathbb{F}_p[X_1,\ldots,X_n]$. Let f be a polynomial in $\mathbb{F}_p[X_1,\ldots,X_n]$. Let $d = 2\cdot max(d_P,d_R,d_T,d_f)$ where d_f is the total degree of f and $d_X = max\{d_f | f \in X\}$ for a set X. If f is independent of (P,R,T), then any algorithm $\mathcal{A}$ that solves the (P,R,T,f)-DDH problem in $\mathbb{G}_T$ with advantage $1/2$ must take at least $\Omega(\sqrt{p/d}-n)$. If $f\cdot R$ is independent of (P,R,T), then the same statement holds for the (P,R,T,f)-DDH problem in $\mathbb{G}$.*

5.2 Analysis of Assumption 1 for (n,ρ,Q,J)

We analyze the Assumption 1 for (n,ρ,Q,J) in the generic group model by using Theorem 5.1. However, we cannot directly apply the theorem to this assumption because the assumption contains negative exponents. To solve this negative exponent problem, we set $\hat{h} = \hat{g}^{\prod_{(i,j)\in Q}(b_i+b_j)}$ and use $\hat{h}$ instead of $\hat{g}$. In this case, the Assumption 1 is described as follows:

$$D = \big(g, g^a, \{g^{b_k}\}_{k=1}^n, \{g^{ab_k}\}_{k\in J}, \hat{h}, \{\hat{h}^{1/(b_i+b_j)}\}_{(i,j)\in Q}\big),\ Z_0 = g^{ab\rho},\ Z_1 = g^d.$$

Let $\eta = \prod_{(i,j)\in Q}(B_i+B_j)$ be a random variable where the maximum degree of η is $n(n-1)/2$. The Assumption 1 is described again as the following sets of multi-variate polynomials:

$$P = \{1, A\} \cup \{B_k\}_{k=1}^n \cup \{AB_k\}_{k\in J},\ R = \{\eta\} \cup \{\eta/(B_i+B_j)\}_{(i,j)\in Q},\ T = \{\},\\ f_0 = AB_\rho,\ f_1 = D.$$

To apply the master theorem, we must show that $f_0\cdot R$ and $f_1\cdot R$ are independent of (P,R,T) by following Definition 5.1. We can easily show that $f_1\cdot R$ is independent of (P,R,T) by using the fact that the random variable D in f_1 does not exist in P,R,T. To show that $f_0\cdot R$ is independent of (P,R,T), we derive two sets $f_0\cdot R$ and $P\cdot R$ as follows:

$$\begin{aligned} f_0\cdot R =& \{\eta AB_\rho\} \cup \{\eta AB_\rho/(B_i+B_j)\}_{(i,j)\in Q},\\ P\cdot R =& \{\eta, \eta A\} \cup \{\eta B_k\}_{1\le k\le n} \cup \{\eta AB_k\}_{k\in J} \cup\\ & \{\eta/(B_i+B_j)\}_{(i,j)\in Q} \cup \{\eta A/(B_i+B_j)\}_{(i,j)\in Q} \cup\\ & \{\eta B_k/(B_i+B_j)\}_{(i,j)\in Q, 1\le k\le n} \cup \{\eta AB_k/(B_i+B_j)\}_{(i,j)\in Q, k\in J}. \end{aligned}$$

The set $f_0 \cdot R$ consists of two component types: ηAB_ρ and $\eta AB_\rho/(B_i+B_j)$. Since ηAB_ρ additionally includes (B_i+B_j) compared to $\eta AB_\rho/(B_i+B_j)$, these component types are independent of each other. Thus, we can analyze ηAB_ρ and $\eta AB_\rho/(B_i+B_j)$ separately.

- First, we show that ηAB_ρ is independent of $P \cdot R$. At this time, since ηAB_ρ includes random variables A and B_ρ, only $\{\eta AB_k\}$ and $\{\eta AB_k/(B_i+B_j)\}$ in $\{e(P_i,R_j)\}$ can have a dependency. However, ηAB_ρ is independent because of $\rho \notin J$.

- Next, we show that $\eta AB_\rho/(B_i+B_j)$ is independent of $P \cdot R$. The subsets of $P \cdot R$ that contain the random variable A are $\{\eta A\}, \{\eta AB_k\}, \{\eta A/(B_i+B_j)\}$, and $\{\eta AB_k/(B_i+B_j)\}$. Here, the subset $\{\eta AB_k\}$ need not be considered because of $\rho \notin J$. The subset $\{\eta A/(B_i+B_j)\}$ does not need to be considered because it does not contain B_ρ. Now using the remaining subsets $\{\eta A = \eta A(B_i+B_j)/(B_i+B_j)\}$ and $\{\eta AB_k/(B_i+B_j)\}$, we may try to compose a linear equation with $\eta AB_\rho/(B_i+B_j)$. Here, the index k cannot be the index ρ because of $\rho \notin J$. Thus the only way to create a linear equation is to derive

$$\frac{\eta AB_\rho}{(B_\rho+B_k)} = \frac{\eta A(B_\rho+B_k)}{(B_\rho+B_k)} - \frac{\eta AB_k}{(B_\rho+B_k)}$$

  when $(\rho,k) \in Q$. To satisfy the above equation, it is required that $k \in J$ when $(\rho,k) \in Q$. However, if $(\rho,k) \in Q$, we have $k \notin J$ according to the definition of J. Thus $\eta AB_\rho/(B_i+B_j)$ is independent because $AB_k \notin P$ when $(\rho,k) \in Q$.

Therefore, we have that $f_0 \cdot R$ is independent of (P,R,T).

5.3 Analysis of Assumption 2 for (n,ρ,Q)

We analyze the Assumption 2 for (n,ρ,Q) in the generic group model by using Theorem 5.1. However, we cannot directly apply the theorem to the assumption because the assumption contains negative exponents. To solve this negative exponent problem, we set $\hat{h} = \hat{g}^{\prod_{(i,j)\in Q}(b_i+b_j)}$ and use $\hat{h}$ instead of $\hat{g}$. In this case, the Assumption 2 is described as follows:

$$D = \big(g, g^a, \{g^{b_i}\}_{i=1}^n, \{g^{ab_k}\}_{1\le k\neq\rho\le n}, \hat{h}, \{\hat{h}^{c_i}\}_{1\le i\neq\rho\le n}, e(g,\hat{h})^{c_\rho}, \{\hat{h}^{c_i/(b_i+b_j)}\}_{(i,j)\in Q}\big),$$
$$Z_0 = e(g,\hat{h})^{ac_\rho},\ Z_1 = e(g,\hat{h})^{d}.$$

Let $\eta = \prod_{(i,j)\in Q}(B_i+B_j)$ be a random variable where the maximum degree of η is $n(n-1)/2$. The Assumption 2 is described again as the following sets of multi-variate polynomials:

$$P = \{1,A\} \cup \{B_k\}_{k=1}^n \cup \{AB_k\}_{1\le k\neq\rho\le n},$$
$$R = \{\eta\} \cup \{\eta C_i\}_{1\le i\neq\rho\le n} \cup \{\eta C_i/(B_i+B_j)\}_{(i,j)\in Q},\ T = \{\eta C_\rho\},$$
$$f_0 = \eta AC_\rho,\ f_1 = \eta D.$$

To apply the master theorem, we must show that the random variables f_0 and f_1 are independent of (P,R,T) by following Definition 5.1. We can easily show that f_1 is independent of (P,R,T) by using the fact that the random variable D in f_1 does not exist in P,R,T. To show that f_0 is independent of (P,R,T),

we derive the set $P \cdot R$ as follows:

$$\begin{aligned} P \cdot R = &\{\eta, \eta A\} \cup \{\eta B_k\}_{i=k}^{n} \cup \{\eta A B_k\}_{1 \le k \ne \rho \le n} \cup \{\eta C_i, \eta A C_i\}_{1 \le i \ne \rho \le n} \cup \\ &\{\eta B_k C_i\}_{1 \le i \ne \rho \le n, 1 \le k \le n} \cup \{\eta A B_k C_i\}_{1 \le i \ne \rho \le n, 1 \le k \le n} \cup \\ &\{\eta C_i/(B_i + B_j)\}_{(i,j) \in Q} \cup \{\eta A C_i/(B_i + B_j)\}_{(i,j) \in Q} \cup \\ &\{\eta B_k C_i/(B_i + B_j)\}_{(i,j) \in Q, 1 \le k \ne \rho \le n} \cup \{\eta A B_k C_i/(B_i + B_j)\}_{(i,j) \in Q, 1 \le k \ne \rho \le n}. \end{aligned}$$

We show that $f_0 = \eta A C_\rho$ is independent of $P \cdot R$ and T. The subsets of $P \cdot R$ that contain the random variables A, C_ρ are $\{\eta A C_i/(B_i + B_j)\}$ and $\{\eta A B_k C_i/(B_i + B_j)\}$. Here, the subset $\{\eta A C_i/(B_i + B_j)\}$ does not need to be considered because it lacks $(B_i + B_j)$. By using the remaining subset $\{\eta A B_k C_i/(B_i + B_j)\}$, we may try to compose a linear equation with $\eta A C_\rho$. The only way to create a linear equation is to derive

$$\eta A C_\rho = \frac{\eta A B_{k_1} C_\rho}{(B_\rho + B_j)} + \frac{\eta A B_{k_2} C_\rho}{(B_\rho + B_j)}$$

when $(\rho, j) \in Q$, $k_1 = \rho$, and $k_2 = j$. To satisfy the above equation, it is required that $k_1 = \rho$ where k_1 is an index for $\{A B_k\}$. However, we have $k_1 \ne \rho$ from the restriction of the Assumption 2. Therefore, f_0 is independent of (P, R, T).

6 Conclusion

From the practical point of view, designing an efficient FE scheme that provides practical functionality is an important issue. In this paper, we newly defined the concept of FE for set intersection (FE-SI) in the multi-client setting and proposed two efficient FE-SI schemes in bilinear groups. The FE-SI scheme issues encryption keys for all clients with a single setup and issues a function key for two client indexes to a third party that wants to perform the set intersection. And we also proposed another FE-SI scheme to add additional control to the function key by limiting the validity of the function key for a specified time period. In addition to this, we have shown that our FE-SI schemes can be extended to provide functionality such as associated message encryption, set intersection cardinality, multi-party set intersection, and forward secrecy.

Because of this research, we have identified interesting problems. The first problem is to prove the security of our FE-SI schemes under simple assumptions. Since the assumptions we have introduced are dynamic assumptions that depend on the function key queries of an attacker, it is necessary to change them to simpler assumptions. The second problem is to prove our FE-SI schemes in a stronger security model than the static security model. In particular, it is necessary to prove the security of FE-SI schemes in a security model that does not requires all function key queries of an attacker in the initial stage. The third problem is to improve the performance of the decryption algorithm of our FE-SI schemes.

References

[1] Apple and google privacy-preserving contact tracing. `https://covid19.apple.com/contacttracing`, 2020.

[2] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[3] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015.

[4] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 601–626. Springer, 2017.

[5] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, volume 9816 of *Lecture Notes in Computer Science*, pages 333–362. Springer, 2016.

[6] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017*, volume 10401 of *Lecture Notes in Computer Science*, pages 67–98. Springer, 2017.

[7] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[9] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[10] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography - TCC 2011*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: A new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

[12] Jan Camenisch, Markulf Kohlweiss, Alfredo Rial, and Caroline Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 196–214. Springer, 2009.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

[14] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM SIGSAC Conference on Computer and Communications Security - CCS 2017*, pages 1243–1255. ACM, 2017.

[15] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018*, volume 11273 of *Lecture Notes in Computer Science*, pages 703–732. Springer, 2018.

[16] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security - CCS'13*, pages 789–800. ACM, 2013.

[17] Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated PSI cardinality with applications to contact tracing. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020*, volume 12493 of *Lecture Notes in Computer Science*, pages 870–899. Springer, 2020.

[18] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography - TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005.

[19] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[20] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 44–61. Springer, 2010.

[21] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[22] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *ACM Symposium on Theory of Computing - STOC 2009*, pages 169–178. ACM, 2009.

[23] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[24] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

[25] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security - CCS 2006*, pages 89–98. ACM, 2006.

[26] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *Network and Distributed System Security Symposium - NDSS 2012*. The Internet Society, 2012.

[27] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In Stuart I. Feldman and Michael P. Wellman, editors, *ACM Conference on Electronic Commerce - EC-99*, pages 78–86. ACM, 1999.

[28] Seny Kamara, Payman Mohassel, Mariana Raykova, and Seyed Saeed Sadeghian. Scaling private set intersection to billion-element sets. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security - FC 2014*, volume 8437 of *Lecture Notes in Computer Science*, pages 195–215. Springer, 2014.

[29] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[30] Kwangsu Lee. Efficient multi-client functional encryption for conjunctive equality and range queries. Cryptology ePrint Archive, Report 2020/822, 2020. `http://eprint.iacr.org/2020/822`.

[31] Kwangsu Lee and Dong Hoon Lee. Two-input functional encryption for inner products from bilinear maps. *IEICE Transactions*, 101-A(6):915–928, 2018.

[32] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 1999.

[33] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[34] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *IEEE Data Eng. Bull.*, 43(2):95–107, 2020.

[35] Tim van de Kamp, David Stritzl, Willem Jonker, and Andreas Peter. Two-client and multi-client functional encryption for set intersection. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy - ACISP 2019*, volume 11547 of *Lecture Notes in Computer Science*, pages 97–115. Springer, 2019.