Registration-Based Functional Encryption

Pratish Datta[1], Tapas Pal[2]

[1] NTT Research, Sunnyvale, CA 94085, USA
`pratish.datta@ntt-research.com`,
[2] NTT Social Informatics Laboratories, Japan 180-8585
`tapas.pal.wh@hco.ntt.co.jp`

Abstract. This paper introduces registered functional encryption (RFE) that eliminates trust on the central authority for handling secrets. Unlike standard functional encryption (FE), in an RFE scheme, users create their secret keys themselves and then register the associated public keys to a key curator along with the functions they wish to compute on the encrypted data. The key curator aggregates the public keys from the different users into a single *compact* master public key. To decrypt, users occasionally need to obtain helper decryption keys from the key curator which they combine with their own secret keys. We require that the size of the aggregated public key, the helper decryption keys, the ciphertexts, as well as the encryption/decryption time to be polylogarithmic in the number of registered users. Moreover, the key curator is entirely *transparent* and maintains no secrets. RFE generalizes the notions of registration-based encryption (RBE) introduced by Garg et al. (TCC 2018) and registered attribute-based encryption introduced by Hohenberger et al. (EUROCRYPT 2023) who dealt with the "all-or-nothing" variants of FE.
We present an RFE scheme for general functions and arbitrary number of users from indistinguishability obfuscation and somewhere statistically binding hash functions. Surprisingly, our construction is achieved via only a minor tweak applied to the registered ABE of Hohenberger et al.

1 Introduction

Functional encryption. *Functional encryption* (FE) [2,10] is an advanced encryption method for computing arbitrary functions on encrypted data. In an FE scheme, there is a central authority which holds a master secret key. The authority generates secret keys sk_f for users. An encryption of a message m can be decrypted using SK_f to recover $f(m)$. FE overcomes the limitation of "all-or-nothing" decryption property of traditional public key encryptions (PKE) by delivering fine-grained access control and computing capability over encrypted data. As a result, various forms of FE have been used to solve challenging cryptographic problems such as constructing indistinguishability obfuscation [3,8,5].

Can we trust the central authority forever? Like all other usual PKE schemes, the current structure of FE scheme has a central authority or a secret key generation center which is responsible for keeping users' secret keys private. In other words, once the central authority is compromised then we can no longer ensure privacy of our encrypted data. This is because an adversary can generate sk_f for any desirable function if it gets hand to the master secret key. Therefore, it is not advisable to trust the central authority forever while our aim is to build a sustainable encryption mechanism that keeps our data safe and secure.

Registration-based encryption. Registration-based encryption (RBE), introduced by Garg et al. [4], is a modern solution to deal with the key escrow problem in the setting of identity-based encryption (IBE) [12,11,1,4]. The secret keys and ciphertexts are associated with identities of receivers and senders respectively. If these two identities match then the receiver is able to decrypt the ciphertext. The role of central authority of IBE is played by a *key curator* in the case of RBE. The key curator, instead of generating secret keys, now *aggregates* independently generated public keys of the users in the system into a *short* master public key MPK. More precisely, a RBE scheme allows users to generate their own public, secret key pairs independently, and then provide only the public keys along with their identities to the key curator. The key curator runs an aggregation algorithm to create a master public key MPK having size much shorter than the number of users in the system. It also generates a *helper decryption key* for each users to facilitate successful decryption. Importantly, the key curator does not process any secret information and it is a *public* algorithm which deterministically computes MPK and the helper decryption keys. We can encrypt data using the current MPK and whenever a new user joins the system the master public key is revised. Consequently, the users need to *update* their helper decryption keys whenever it is necessary for a successful decryption during the life-time of the system. If there are L users in the system then each each user is required to update the helper decryption key at most $O(\log L)$ times. Moreover, the sizes of MPK and the helper decryption keys remain at most $O(\log L)$.

Recently, Hohenberger et al. [7] extends the notion of RBE to *registered attribute-based encryption* (RABE) with the motivation to solve the key escrow problem in ABE [6,9]. In RABE, the aggregation of public keys are performed with respect to user specific sets of attributes and encryption is done under some policies. At the time of decryption, a user recovers the message if it's set of attributes satisfies the policy. Although IBE and ABE are a type of FE, both of these falls under "all-or-nothing" encryption mechanism where a successful decryption reveals all information about the data. This led us to the following open problem.

Open Problem *Can we construct a registration-based FE for general circuits?*

1.1 Our Results

In this paper, we introduce notion of registered functional encryption (RFE) to resolve the key escrow problem for FE and supporting the arbitrary number of users. This notion generalizes registration-based encryption (RBE) introduced by Garg et al. [4] and registered Attribute-based encryption (RABE) introduced by Hohenberger et al. [7] both of which dealt with "all-or-noting" encryption paradigms. Switching from all-or-nothing to the setting of FE seems to present additional challenges. However, we show that a minor tweak to the registered ABE of [7] could lead to an RFE scheme. Indeed, we present an RFE scheme for general functions and an arbitrary number of users from indistinguishability obfuscation ($i\mathcal{O}$) and somewhere statistically binding (SSB) hash functions.

While this is mostly a feasibility result, it gives us some insights into the gap between registered ABE and RFE. Indeed, it is often seen that constructing a standard FE scheme for general functions (supporting arbitrary collusion of users) is much more difficult compared to ABE. Our results show that this is not the case for RFE.

2 Cryptographic tools

Definition 1 (Pseudorandom Generator) A pseudorandom generator (PRG) $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+\ell(\lambda)}$ with stretch $\ell(\lambda)$ (ℓ is some polynomial function) is a polynomial-time computable function that satisfies the following. For any PPT adversary $\mathcal{A}$, it holds that

$$|\Pr[\mathcal{A}(\mathsf{PRG}(s)) = 1 : s \leftarrow \{0,1\}^{\lambda}] - \Pr[\mathcal{A}(r) : r \leftarrow \{0,1\}^{\lambda+\ell(\lambda)}] \leq \mathsf{negl}(\lambda)$$

.

Definition 2 (Secret Key Encryption) Let λ be a security parameter and let p, q, r and s be some polynomials. A secret key encryption scheme is a tuple of algorithms $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M} \coloneqq \{0,1\}^n$, ciphertext space $\mathcal{C} \coloneqq \{0,1\}^{\ell_c(\lambda)}$, and secret key space $\mathcal{SK} \coloneqq \{0,1\}^{\ell_k(\lambda)}$.

$\mathsf{Setup}(1^{\lambda})$: The setup algorithm takes the security parameter 1^{λ} as input and outputs a secret key $\mathsf{sk} \in \mathcal{SK}$.

$\mathsf{Enc}(\mathsf{sk}, m)$: The encryption algorithm takes sk and a plaintext $m \in \mathcal{M}$ as input, and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes sk and ct as input, and outputs a plaintext $m' \in \mathcal{M}$ or $\perp$.

The algorithms must satisfy the following properties:

Correctness: There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq m \ : \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{Setup}(1^{\lambda}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Security: Let $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme. We consider the following security experiment $\mathsf{Expt}^{\mathsf{SKE}}_{\mathcal{A}}(\lambda, b)$ against a PPT adversary $\mathcal{A}$.

1. The challenger computes $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and returns ct to $\mathcal{A}$. $\mathcal{A}$ can repeat this process polynomially many times.
3. $\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.
4. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$ and sends ct to $\mathcal{A}$.
5. $\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and returns ct to $\mathcal{A}$. $\mathcal{A}$ can repeat this process polynomially many times.
6. $\mathcal{A}$ outputs $b' \in \{0,1\}$. This is the output of the experiment.

We say that SKE is IND-CPA secure if, for any PPT $\mathcal{A}$, it holds that

$$|\Pr[\mathsf{Expt}^{\mathsf{SKE}}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{SKE}}_{\mathcal{A}}(\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda).$$

Definition 3 (Indistinguishability Obfuscator) A PPT algorithm $i\mathcal{O}$ is a secure IO for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following two conditions.

- **Functionality:** For any security parameter $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}_\lambda$, and input x, we have that
$$\Pr[C'(x) = C(x) \mid C' \leftarrow i\mathcal{O}(1^\lambda, C)] = 1 \ .$$
- **Indistinguishability:** For any pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ satisfying $C_0(x) = C_1(x), \forall x$ and any PPT distinguisher $\mathcal{D}$, the following holds:
$$\left|\Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_0)) = 1\right] - \Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_1)) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 4 (Somewhere Statistically Binding Hash Function) Let λ be a security parameter. A somewhere statistically binding (SSB) hash function with block length $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$, output length $\ell_{\mathsf{hash}} = \ell_{\mathsf{hash}}(\lambda)$, and opening length $\ell_{\mathsf{open}} = \ell_{\mathsf{open}}(\lambda)$ is a tuple of efficient algorithms $\mathsf{SSB} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Vrfy})$ with the following properties:

$\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, N, i^*)$: The setup algorithm takes as input a security parameter λ, a block size ℓ_{blk}, the message length $N \leq 2^\lambda$, and an index $i^* \in [N]$, and outputs a hash key hk. Both N and i^* are encoded in *binary*; in particular, this means that $|\mathsf{hk}| = \mathsf{poly}(\lambda, \ell_{\mathsf{blk}}, \log N)$. We let $\Sigma = \{0,1\}^{\ell_{\mathsf{blk}}}$ denote the block alphabet.

$\mathsf{Hash}(\mathsf{hk}, x)$: the hash algorithm takes as input a hash key hk and input x, and outputs a hash value $h \in \{0,1\}^{\ell_{\mathsf{hash}}}$.

$\mathsf{Open}(\mathsf{hk}, \mathbf{x}, i)$: The open algorithm takes as input a hash key hk, an input $\mathbf{x} \in \Sigma^N$ and an index i, and outputs an opening $\pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$.

$\mathsf{Vrfy}(\mathsf{hk}, h, i, x_i, \pi_i)$: The verify algorithm takes as input a hash key hk, a hash value h, an index i, a value $x_i \in \Sigma$, and an opening $\pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$, and outputs a bit $b \in \{0,1\}$ indicating whether it accepts or rejects.

The algorithm must satisfy the following properties:

Correctness: For all security parameter $\lambda \in \mathbb{N}$, all block sizes $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$, all integers $N \leq 2^{\lambda}$, all indices $i, i^* \in [N]$, and any $\mathbf{x} \in \Sigma^N$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{hk}, h, i, x_i, \pi_i) = 1 : \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell_{\mathsf{blk}}}, N, i^*) \\ h \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}); \pi_i \leftarrow \mathsf{Open}(\mathsf{hk}, \mathbf{x}, i) \end{array}\right] = 1.$$

• **Index hiding**: For a bit $b \in \{0,1\}$ and an adversary $\mathcal{A}$, define the index hiding experiment $\mathsf{Expt}^{\mathsf{indexSSB}}_{\mathcal{A}}(1^{\lambda}, b)$ as follows:

1. $\mathcal{A}$ chooses an integer N and two indices $i_0, i_1 \in [N]$.
2. The challenger samples $\mathsf{hk} \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell_{\mathsf{blk}}}, N, i_b)$ and gives hk to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is also the output of the experiment.

We require that for all polynomials $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$ and for all efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathsf{Expt}^{\mathsf{indexSSB}}_{\mathcal{A}}(1^{\lambda}, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{indexSSB}}_{\mathcal{A}}(1^{\lambda}, 1) = 1]| = \mathsf{negl}(\lambda).$$

• **Somewhere statistically binding**: We say that a hash key hk is statistically binding for an index $i^* \in [N]$ if there does not exists $h \in \{0,1\}^{\ell_{\mathsf{hash}}}$, $x \neq x' \in \Sigma$, and π, π' where $\mathsf{Vrfy}(\mathsf{hk}, h, i^*, x, \pi) = 1 = \mathsf{Vrfy}(\mathsf{hk}, h, i^*, x', \pi')$. We require that for all polynomial $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$ and for all $N \leq 2^{\lambda}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all $i \in [N]$,

$$\Pr[\mathsf{hk} \text{ is statistically binding for index } i : \mathsf{hk} \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell_{\mathsf{blk}}}, N, i)] = 1 - \mathsf{negl}(\lambda).$$

• **Succinctness**: The hash length ℓ_{hash}, and opening length ℓ_{open} are all fixed polynomials in the security parameter λ and block size ℓ_{blk} (and independent of N).

3 Registered Functional Encryption

In this section, we introduce the notion of registered FE scheme.

Definition 5 (Registered Functional Encryption) Let $\mathcal{U}_F = \{\mathcal{F}_{\lambda}\}_{\lambda \in \mathbb{N}}$ be the universe of functions and $\mathcal{M}$ be the set of messages. A registered functional encryption scheme with function universe $\mathcal{U}_F$ and message space $\mathcal{M}$ is a tuple of efficient algorithms $\mathsf{RFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{RegPK}, \mathsf{Enc}, \mathsf{Update}, \mathsf{Dec})$ that work as follows:

$\mathsf{Setup}(1^{\lambda}, 1^{\ell_f})$: The setup algorithm takes the security parameter λ, the (maximum) size ℓ_f of the functions in $\mathcal{U}_F$ as inputs and outputs a common reference string crs.

$\mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux})$: The key generation algorithm takes the common reference string crs, and a (possibly empty) state aux as inputs and outputs a public key pk and a secret key sk.

$\mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f_{\mathsf{pk}})$: The registration algorithm takes a common reference string crs, a (possibly empty) state aux, a public key pk and a function $f_{\mathsf{pk}} \in \mathcal{F}_{\lambda}$ as inputs and outputs a master public key MPK and an updated state aux'. This is a *deterministic* algorithm.

$\mathsf{Enc}(\mathsf{MPK}, m)$: The encryption algorithm takes a master public key MPK and a message $m \in \mathcal{M}$ as inputs and outputs a ciphertext ct.

$\mathbf{Update(crs, aux, pk)}$: The update algorithm takes a common reference string crs, a state aux and a public key pk as inputs, and outputs a helper decryption keys hsk. This is a *deterministic* algorithm.

$\mathbf{Dec(sk, hsk, ct)}$: The decryption algorithm takes a secret key sk, a helper decryption key hsk and ciphertext ct as inputs and outputs a message m'. This is a *deterministic* algorithm.
The algorithms must satisfy the following properties:

Correctness and efficiency: For all security parameters $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}$, all functions $f \in \mathcal{F}_\lambda$, we define the following experiment between an adversary $\mathcal{A}$ and a challenger:

- **Setup phase:** The challenger starts by sampling the common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_f})$. It then initializes the auxiliary input $\mathsf{aux} \leftarrow \perp$ and initial master public key $\mathsf{MPK}_0 \leftarrow \perp$. It also initializes a counter $\mathsf{ctr}[\mathsf{reg}] \leftarrow 0$ to keep track of the number of registration queries and another counter $\mathsf{ctr}[\mathsf{enc}] \leftarrow 0$ to keep track of the number of encryption queries. Finally, it initializes $\mathsf{ctr}[\mathsf{reg}]^* \leftarrow \infty$ as the index for the target key. It gives crs to $\mathcal{A}$.
- **Query phase:** During the query phase, the adversary $\mathcal{A}$ is able to make the following queries:
  - **Register non-target key query:** In a non-target-key registration query, the adversary $\mathcal{A}$ specifies a public key pk and a function $f \in \mathcal{U}_F$. The challenger first increments the counter $\mathsf{ctr}[\mathsf{reg}] \leftarrow \mathsf{ctr}[\mathsf{reg}] + 1$ and then registers the key by computing $(\mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}],\mathsf{aux}'}) \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f)$. The challenger updates its auxiliary data by setting $\mathsf{aux} \leftarrow \mathsf{aux}'$ and replies $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{reg}], \mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux})$.
  - **Register target key query:** In a target-key registration query, the adversary specifies a function $f^* \in \mathcal{U}_F$. If the adversary has previously made a target-key registration query, then the challenger replies with $\perp$. Otherwise, the challenger increments the counter $\mathsf{ctr}[\mathsf{reg}] \leftarrow \mathsf{ctr}[\mathsf{reg}] + 1$, samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{aux})$, and registers $(\mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}],\mathsf{aux}'}) \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*, f^*)$. It computes the helper decryption key $\mathsf{hsk}^*\mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$. The challenger updates its auxiliary data by setting $\mathsf{aux} \leftarrow \mathsf{aux}'$, stores the index of the target identity $\mathsf{ctr}[\mathsf{reg}]^* \leftarrow \mathsf{ctr}[\mathsf{reg}]$, and replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{reg}], \mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux}, \mathsf{pk}^*, \mathsf{hsk}, \mathsf{sk}^*)$.
  - **Encryption query:** In an encryption query, the adversary submits the index $\mathsf{ctr}[\mathsf{reg}]^* \leq i \leq \mathsf{ctr}[\mathsf{reg}]$ of a public key and a message $m_{\mathsf{ctr}[\mathsf{enc}]} \in \mathcal{M}$. If the adversary has not yet registered a target key the challenger replies with $\perp$. Otherwise, the challenger increments the counter $\mathsf{ctr}[\mathsf{enc}] \leftarrow \mathsf{ctr}[\mathsf{enc}] + 1$ and computes $\mathsf{ct}_{\mathsf{ctr}[\mathsf{enc}]} \leftarrow \mathsf{Enc}(\mathsf{MPK}_i, m)$. The challenger replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{enc}], \mathsf{ct}_{\mathsf{ctr}[\mathsf{enc}]})$.
  - **Decryption query:** In a decryption query, the adversary submits a ciphertext index $1 \leq j \leq \mathsf{ctr}[\mathsf{enc}]$. The challenger computes $m'_j\mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$. If $m'_j = \mathsf{GetUpdate}$, then the challenger computes an updated helper decryption key $\mathsf{hsk}^* \leftarrow \mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$ and recomputes $m'_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$. If $m'_j \neq f^*(m_j)$, the experiment halts with outputs $b = 1$.

  If $\mathcal{A}$ has finished making queries and the experiment has not halted (as a result of a decryption query), then the experiment outputs $b = 0$.

We say that RFE is correct and efficient if for all adversaries $\mathcal{A}$ making at most polynomial number of queries, the following properties hold:

- **Correctness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the above experiment. We say that the scheme satisfies *perfect correctness* if $\Pr[b = 1] = 0$.

- **Compactness:** Let N be the number of registration queries the adversary makes in the above experiment. There exists a universal polynomial $\mathsf{poly}(\cdot,\cdot,\cdot)$ such that for $i \in [N]$, $|\mathsf{MPK}_i| = \mathsf{poly}(\lambda, \ell_f, \log i)$. We also require that the size of the helper decryption key hsk^* satisfy $\mathsf{hsk}^* = \mathsf{poly}(\lambda, \ell_f, \log N)$ (at *all* point of the experiment).
- **Update efficiency:** Let N be the number of registration queries made by $\mathcal{A}$. Then, in the course of the above experiment, the challenger invokes the update algorithm Update at most $O(\log N)$ times where each invocation runs in $\mathsf{poly}(\log N)$ time in the RAM model fo computation. Specially, we model Update as a RAM program that has *random* access to its input; thus, the running time of Update in the RAM model can be *smaller* than the input length.

Security: Let $b \in \{0,1\}$ be a bit. We define the following security experiment $\mathsf{Expt}^{\mathsf{RFE}}_{\mathcal{A}}(1^\lambda, b)$ played between an adversary $\mathcal{A}$ and a challenger.

- **Setup phase:** The challenger samples a common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_f})$. It then initializes the auxiliary input $\mathsf{aux} \leftarrow \bot$, the initial master public key $\mathsf{MPK} \leftarrow \bot$, a counter $\mathsf{ctr} \leftarrow 0$ for the number of honest-key-registration queries the adversary has made, an empty set of keys $\mathsf{Cor} \leftarrow \emptyset$, and an empty dictionary mapping public keys to registered function $\mathsf{D} \leftarrow \emptyset$. For notational convenience, if $\mathsf{pk} \notin \mathsf{D}$, then we define $\mathsf{D}[\mathsf{pk}] \coloneqq \emptyset$. The challenger gives the crs to $\mathcal{A}$.
- **Query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Registered corrupted key query:** In a corrupted key query, $\mathcal{A}$ specifies a public key pk and a function $f \in \mathcal{U}_F$. The challenger registers the key by computing $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f)$. The challenger updates its copy of the public key $\mathsf{MPK} \leftarrow \mathsf{MPK}'$, its auxiliary data $\mathsf{aux} \leftarrow \mathsf{aux}'$, adds pk to Cor, and updates $\mathsf{D}[\mathsf{pk}] \leftarrow \mathsf{D}[\mathsf{pk}] \cup \{f\}$. It replies to $\mathcal{A}$ with $(\mathsf{MPK}', \mathsf{aux}')$.
  - **Registered honest key query:** In a honest key query, $\mathcal{A}$ specifies a function $f \in \mathcal{U}_F$. The challenger increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$ and samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux})$, and registers the key by computing $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}_{\mathsf{ctr}}, f)$. The challenger updates its public key $\mathsf{MPK} \leftarrow \mathsf{MPK}'$, its auxiliary data $\mathsf{aux} \leftarrow \mathsf{aux}'$, adds $\mathsf{D}[\mathsf{pk}_{\mathsf{ctr}}] \leftarrow \mathsf{D}[\mathsf{pk}_{\mathsf{ctr}}] \cup \{f\}$. It replies to $\mathcal{A}$ with $(\mathsf{ctr}, \mathsf{MPK}', \mathsf{aux}', \mathsf{pk}_{\mathsf{ctr}})$.
  - **Corrupt honest query:** In a corrupt-honest key query, $\mathcal{A}$ specifies an index $1 \leq i \leq \mathsf{ctr}$. Let $(\mathsf{pk}_i, \mathsf{sk}_i)$ be the i-th public/secret key the challenger samples when responding to the i-th honest-key-registration query. The challenger adds pk_i to Cor and replies to $\mathcal{A}$ with sk_i.
- **Challenge phase:** The adversary $\mathcal{A}$ chooses two messages $m_0^*, m_1^* \in \mathcal{M}$. The challenger replies with the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{MPK}, m_b^*)$.
- **Output phase:** At the end of the experiment, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

Let $\mathcal{S} = \{f_{\mathsf{pk}} \in \mathsf{D}[\mathsf{pk}] : \mathsf{pk} \in \mathsf{Cor}\}$. We say an adversary $\mathcal{A}$ is admissible if for all corrupted slot indices $f_{\mathsf{pk}} \in \mathsf{Cor}$, it holds that $f_{\mathsf{pk}}(m_0^*) = f_{\mathsf{pk}}(m_1^*)$. The registration-based functional encryption scheme RFE is said to be secure if for all admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathsf{Expt}^{\mathsf{RFE}}_{\mathcal{A}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{RFE}}_{\mathcal{A}}(1^\lambda, 1) = 1]| = \mathsf{negl}(\lambda).$$

4 Slotted Registered Functional Encryption

In this section, we introduce the notion of slotted registered FE scheme.

Definition 6 (Slotted Registered Functional Encryption) Let $\mathcal{U}_F = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be the universe of functions and $\mathcal{M}$ be the set of messages. A slotted registered functional encryption scheme with function universe $\mathcal{U}_F$ and message space $\mathcal{M}$ is a tuple of efficient algorithms $\mathsf{SlotRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ that work as follows:

$\mathbf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L)$: The setup algorithm takes the security parameter λ, the (maximum) size $|\mathcal{U}_F|$ of the functions in $\mathcal{U}_F$ and the number of slots L (in unary) as inputs and outputs a common reference string crs.

$\mathbf{KeyGen}(\mathbf{crs}, i)$: The key generation algorithm takes the common reference string crs, and a slot index $i \in [L]$ as inputs and outputs a public key pk_i and a secret key sk_i for the slot i.

$\mathbf{IsValid}(\mathbf{crs}, i, \mathbf{pk}_i)$: The key-validation algorithm takes a common reference string crs, a slot index $i \in [L]$ and a public key pk_i as inputs and outputs a bit $b \in \{0, 1\}$. This is a *deterministic* algorithm.

$\mathbf{Aggregate}(\mathbf{crs}, (\mathbf{pk}_1, f_1), \ldots, (\mathbf{pk}_L, f_L))$: The aggregate algorithm takes a common reference string crs, a list of L public key-function pairs $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ as inputs such that $f_i \in \mathcal{F}_\lambda$ for all $i \in [L]$ and outputs a master public key MPK and a collection of helper decryption keys $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$. This is a *deterministic* algorithm.

$\mathbf{Enc}(\mathbf{MPK}, m)$: The encryption algorithm takes a master public key MPK and a message $m \in \mathcal{M}$ as inputs and outputs a ciphertext ct.

$\mathbf{Dec}(\mathbf{sk}, \mathbf{hsk}, \mathbf{ct})$: The decryption algorithm takes a secret key sk, a helper decryption key hsk and ciphertext ct as inputs and outputs a message m'. This is a *deterministic* algorithm.
The algorithms must satisfy the following properties:

Completeness: For all $\lambda \in \mathbb{N}$, all property universes $\mathcal{U}_P$, and all indices $i \in [L]$,

$$\Pr\left[\mathsf{IsValid}(\mathsf{crs}, i, \mathsf{pk}_i) = 1 : \ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i)\right] = 1.$$

Correctness: The $\mathsf{SlotRFE}$ is said to be correct if for all security parameters $\lambda \in \mathbb{N}$, all possible lengths $L \in \mathbb{N}$, all indices $i \in [L]$, if we sample $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|, 1^L})$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i)$ and for all collections of public keys $\{\mathsf{pk}_j\}_{j \neq i}$ (which may be correlated to pk_i) where $\mathsf{IsValid}(\mathsf{crs}, j, \mathsf{pk}_j) = 1$, all messages $m \in \mathcal{M}$, all functions $f \in \mathcal{F}_\lambda$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}) = f(m) : \begin{array}{c} (\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L) \leftarrow \mathsf{Aggregate}(\mathsf{MPK}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)); \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m) \end{array}\right] = 1.$$

Compactness: The $\mathsf{SlotRFE}$ is said to be compact if there exists a universal polynomial $\mathsf{poly}(\cdot, \cdot, \cdot)$ such that the length of the master public key and individual helper secret keys output by $\mathsf{Aggregate}$ are bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$.

Security: Let $b \in \{0,1\}$ be a bit. We define the following security experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, b)$ played between an adversary $\mathcal{A}$ and a challenger.

- **Setup phase:** The adversary sends a slot count 1^L to the challenger. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L)$ and sends crs to $\mathcal{A}$. The challenger initializes a counter $\mathsf{ctr} \leftarrow 0$, a dictionary D and a set of slot indices Cor.
- **Pre-challenge query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Key-generation query:** In a key-generation query, $\mathcal{A}$ specifies a slot index $i \in [L]$. The challenger samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i)$ and increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$. Then, it sends $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}})$ to $\mathcal{A}$. The challenger adds the mapping $\mathsf{ctr} \mapsto (i, \mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}})$ to the dictionary D.
  - **Corruption query:** In a corruption query, $\mathcal{A}$ specifies an index $1 \leq c \leq \mathsf{ctr}$. The challenger looks up the tuple $(i', \mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and sends sk' to $\mathcal{A}$.
- **Challenge phase:** For each slot $i \in [L]$, $\mathcal{A}$ specifies a tuple (c_i, pk_i^*) where either $c_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $c_i = \perp$ to reference a key outside this set. $\mathcal{A}$ also specifies two challenge messages m_0^*, m_1^*. The challenger does the following:
  - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', \mathsf{pk}', \mathsf{sk}')$. If $i = i'$, then the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}'$. Moreover, if $\mathcal{A}$ previously issues a *corruption query* on the index c_i, then the challenger adds the slot index i to Cor. Otherwise, if $i \neq i'$, then the experiment halts.
  - If $c_i = \perp$, then the challenger checks $\mathsf{IsValid}(\mathsf{crs}, i, \mathsf{pk}_i^*) = 1$. If not, the experiment halts. If the key is valid, the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}_i^*$ and adds the slot index i to Cor.

  The challenger computes $(\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L) \leftarrow \mathsf{Aggregate}(\mathsf{MPK}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ and then $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{MPK}, m_b^*)$. Finally, it sends ct^* to $\mathcal{A}$. Note that, there is no need to additionally provide $(\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L)$ to $\mathcal{A}$ since $\mathsf{Aggregate}$ is a deterministic algorithm. Similarly, there is no advantage of allowing $\mathcal{A}$ to select the challenge messages after seeing the aggregated key.
- **Post-challenge query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - In a corruption query, $\mathcal{A}$ specifies a slot index $c \in \{1, \ldots, \mathsf{ctr}\}$. The challenger picks the tuple $(i', \mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and sends sk' to $\mathcal{A}$. Moreover, if $\mathcal{A}$ registered a tuple of the form (c, pk^*) in the challenge phase for some choice of pk^*, then the challenger adds the slot index $i' \in [L]$ to Cor.
- **Output phase:** At the end of the experiment, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

We say an adversary $\mathcal{A}$ is admissible if for all corrupted slot indices $i \in \mathsf{Cor}$, it holds that $f(m_0^*) = f(m_1^*)$. The slotted registration-based encryption scheme $\mathsf{SlotRFE}$ is said to be secure if for all polynomials $L = L(\lambda)$ and all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, 1) = 1]| = \mathsf{negl}(\lambda).$$

Remark 1 The security definition above allows the adversary to make additional corruption queries in a post-challenge query phase. However, as shown in [7], the security in the setting without post-challenge queries implies the security in the setting with post-challenge queries since the aggregation algorithm is *deterministic*. Hence, we only consider slotted registered FE with a security notion that does not involve any post-challenge queries.

5 Slotted Registered FE from Indistinguishability Obfuscation

Construction: We use the following cryptographic tools as building blocks:

- A length doubling PRG $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$.
- A secret key encryption scheme $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$.
- A somewhere statistically binding hash function $\mathsf{SSB} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Vrfy})$.
- An indistinguishability obfuscation $i\mathcal{O}$ for $\mathsf{P/poly}$.

The slotted registered functional encryption $\mathsf{SlotRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ for a function universe $\mathcal{U}_F = \{0,1\}^{\ell_f}$, and message space $\mathcal{M}$ works as follows:

$\mathsf{Setup}(1^\lambda, 1^{\ell_f}, L)$: The setup algorithm takes the security parameter λ, the bit-length ℓ_f of a function in $\mathcal{U}_F$ (in unary) and the number of users L (in binary) as inputs and sets $\ell_{\mathsf{blk}} = \ell_f + 2\lambda$, computes $\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, 1)$ and sets $\mathsf{crs} \coloneqq \mathsf{hk}$. It outputs crs.

$\mathsf{KeyGen}(\mathsf{crs}, i)$: The key generation algorithm takes the common reference string crs, and a slot index $i \in [L]$ as inputs and samples $s_i \leftarrow \{0,1\}^\lambda$. It outputs the public key as $\mathsf{pk}_i \coloneqq \mathsf{PRG}(s_i)$ and the secret key as $\mathsf{sk}_i \coloneqq s_i$.

$\mathsf{IsValid}(\mathsf{crs}, i, \mathsf{pk}_i)$: The key-validation algorithm takes a common reference string crs, a slot index $i \in [L]$ and a public key pk_i as inputs and outputs 1 if $\mathsf{pk}_i \in \{0,1\}^{2\lambda}$; otherwise outputs 0.

$\mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$: The aggregate algorithm takes a common reference string crs, a list of L public key-function pairs $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ as inputs such that $f_i \in \mathcal{U}_F$ for all $i \in [L]$. It computes

$$h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$$

and sets $\mathsf{MPK} \coloneqq (\mathsf{hk}, h)$. For each user $i \in [L]$, the aggregate algorithm computes

$$\pi_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)), i)$$

where we treat each pair $(\mathsf{pk}_i, f_i) \in \{0,1\}^{\ell_{\mathsf{blk}}}$ as one SSB hash-block. It sets the helper decryption key as $\mathsf{hsk}_i \coloneqq (i, \mathsf{pk}_i, f_i, \pi_i)$ and outputs $\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$.

$\mathsf{Enc}(\mathsf{MPK}, m)$: The encryption algorithm takes a master public key MPK, and a message $m \in \mathcal{M}$ as inputs and samples $\mathsf{SK}_0, \mathsf{SK}_1 \leftarrow \mathsf{SKE.Setup}(1^\lambda)$, computes

$$\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m) \text{ and } \mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m|}).$$

It writes $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. The algorithm samples $u_{k,\beta} \leftarrow \{0,1\}^\lambda$ for all $k \in [2\ell_c], \beta \in \{0,1\}$. It computes $V = (v_{k,\beta} \coloneqq \mathsf{PRG}(u_{k,\beta}))_{k \in [2\ell_c], \beta \in \{0,1\}}$. It constructs the circuit $C_0 = C[\mathsf{MPK}, \mathsf{SK}_0, V]$ as defined in Figure 1 and computes $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$. It outputs the ciphertext $\mathsf{ct} \coloneqq (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}} \coloneqq (u_{k,\beta_k})_{k \in [2\ell_c]})$.

$\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct})$: The decryption algorithm takes a secret key sk_i, a helper decryption key $\mathsf{hsk}_i = (i, \mathsf{pk}_i, f_i, \pi_i)$ and ciphertext $\mathsf{ct} = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}})$ as inputs and outputs $\widetilde{C}_0(\mathsf{sk}_i, i, \mathsf{pk}_i, f_i, \pi_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$.

Constants: $\mathsf{MPK} = (\mathsf{hk}, h), \mathsf{SK}_j, V = (v_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}$
Inputs: $\mathsf{sk}_i \in \{0,1\}^\lambda, i \in [L], \mathsf{pk}_i \in \{0,1\}^{2\lambda}, f_i \in \{0,1\}^{\ell_f}, \pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$, SKE ciphertexts $\{\mathsf{CT}_j\}_{j\in\{0,1\}}$ and $\sigma_{\mathsf{CT}} = (u_k)_{k\in[2\ell_c]}$

```
1. Parse $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$.
2. If $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{pk}_i, f_i), \pi_i) = 1 \wedge \mathsf{PRG}(\mathsf{sk}_i) = \mathsf{pk}_i \wedge (\mathsf{PRG}(u_k) = v_{k,\beta_k})_{k\in[2\ell_c]}$
    a. Compute $\widehat{m} \leftarrow \mathsf{SKE.Dec}(\mathsf{SK}_j, \mathsf{CT}_j)$
    b. Output $f_i(\widehat{m})$
3. Else, output $\perp$
```

Fig. 1: The circuit $C_j = C_j[\mathsf{MPK}, \mathsf{SK}_j, V]$ for $j \in \{0,1\}$

Completeness: The scheme satisfies completeness since the IsValid algorithm outputs 1 only if $\mathsf{pk}_i \in \{0,1\}^{2\lambda}$ and the KeyGen algorithm computes $\mathsf{pk}_i = \mathsf{PRG}(s_i)$ which belongs to $\{0,1\}^{2\lambda}$.

Correctness: Consider a secret key $\mathsf{sk}_i = s_i$, a helper decryption key $\mathsf{hsk}_i = (i, \mathsf{pk}_i, f_i, \pi_i)$, and a ciphertext $\mathsf{ct} = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}})$ generated as above. By definition, $\mathsf{pk}_i = \mathsf{PRG}(\mathsf{sk}_i)$ and $\mathsf{MPK} = (\mathsf{hk}, h)$ where

$$h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$$
$$\pi_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)), i).$$

Therefore, the check of the circuit C_0 passes, i.e., $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{pk}_i, f_i), \pi_i) = 1$ and $\mathsf{PRG}(\mathsf{sk}_i) = \mathsf{pk}_i$ by the correctness of SSB and PRG. Also, by definition $\mathsf{PRG}(u_k) = v_{k,\beta_k}$ holds for all $k \in [2\ell_c]$ where $\sigma_{\mathsf{CT}} = (u_k)_{k\in[2\ell_c]}$. Then, the SKE decryption $\mathsf{SKE.Dec}(\mathsf{SK}_0, \mathsf{CT}_0)$ returns m, and hence the circuit C_0 on input $(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$ returns $f_i(m)$. Therefore, by the correctness of $i\mathcal{O}$, we get $\widetilde{C}_0(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}}) = f_i(m)$.

Compactness: Consider the master public key $\mathsf{MPK} = (\mathsf{hk}, h)$ and the helper decryption key $\mathsf{hsk}_i = (i, \mathsf{pk}_i, f_i, \pi_i)$ output by the Aggregate algorithm. Since SSB.Setup is an efficient algorithm we have $|\mathsf{hk}| = \mathsf{poly}(\lambda, \ell_{\mathsf{blk}}, \log L)$ and due to the succinctness of SSB.Hash we have $|h|, |\pi_i| = \mathsf{poly}(\lambda, \ell_{\mathsf{blk}})$. The maximum length of any function in the universe $\mathcal{U}_F$ is ℓ_f and $\ell_{\mathsf{blk}} = \ell_f + 2\lambda = \log(|\mathcal{U}_F|) + 2\lambda$. Therefore, it must hold that $|\mathsf{MPK}|, |\mathsf{hsk}_i|$ are bounded by $\mathsf{poly}(\lambda, \log(|\mathcal{U}_F|), \log L)$.

Security: We prove the following theorem to show that the SlotRFE is secure.

Theorem 1 *Assuming that the* PRG *is secure,* SKE *is IND-CPA secure,* SSB *is correct and secure, and $i\mathcal{O}$ is secure then our* SlotRFE *is secure.*

Proof. We prove the theorem using a sequence of hybrid experiments. We start with a real experiment which is $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, 0)$ and end up in $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, 1)$. The computational indistinguishability between the consecutive hybrids will be argued based on the assumptions stated in the theorem.

$\underline{\mathsf{Hybd}_0}$: This the real experiment with $b = 0$. More precisely, it works as follows:

- **Setup phase:** The adversary sends a slot count 1^L to the challenger. The challenger samples $\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, 1)$ and sends $\mathsf{crs} \coloneqq \mathsf{hk}$ to $\mathcal{A}$. The challenger initializes a counter $\mathsf{ctr} \leftarrow 0$, a dictionary D and a set of slot indices Cor.

- **Pre-challenge query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Key-generation query:** In a key-generation query, $\mathcal{A}$ specifies a slot index $i \in [L]$. The challenger samples $s \leftarrow \{0,1\}^\lambda$ and increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$. Then, it sends $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}} \coloneqq \mathsf{PRG} s)$ to $\mathcal{A}$. The challenger adds the mapping $\mathsf{ctr} \mapsto (i, \mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}} \coloneqq s)$ to D.
  - **Corruption query:** In a corruption query, $\mathcal{A}$ specifies an index $1 \le c \le \mathsf{ctr}$. The challenger looks up the tuple $(i, \mathsf{pk}, s) \leftarrow \mathsf{D}[c]$ and sends s to $\mathcal{A}$.
- **Challenge phase:** For each slot $i \in [L]$, $\mathcal{A}$ specifies a tuple $(c_i, f_i, \mathsf{pk}_i^*)$, and two challenge messages m_0^*, m_1^*. The challenger does the following:
  - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', \mathsf{pk}', \mathsf{sk}')$. If $i = i'$, then the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}'$. Moreover, if $\mathcal{A}$ previously issues a *corruption query* on the index c_i, then the challenger adds the slot index i to Cor. Otherwise, if $i \neq i'$, then the experiment halts.
  - If $c_i = \perp$, then the challenger checks $\mathsf{pk}_i^* \in \{0,1\}^{2\lambda}$. If not, the experiment halts. Otherwise, the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}_i^*$ and adds the slot index i to Cor.

  The challenger computes $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$ and samples $\mathsf{SK}_0, \mathsf{SK}_1 \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. Then, it computes $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m_0^*)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m_0^*|})$ and $V = (v_{k,\beta}^* \coloneqq \mathsf{PRG}(u_{k,\beta}^*))_{k \in [2\ell_c], \beta \in \{0,1\}}$ where $u_{k,\beta}^* \leftarrow \{0,1\}^\lambda$. Then, it computes $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$ and sets $\sigma_{\mathsf{CT}}^* = (u_{k,\beta_k}^*)_{k \in [2\ell_c]}$ where $C_0 = C[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ (as defined in Figure 1) and β_k represents the k-th bit of $(\mathsf{CT}_0, \mathsf{CT}_1)$. Finally, it sends $\mathsf{ct}^* \coloneqq (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}}^*)$ to $\mathcal{A}$.
- **Output phase:** At the end of the experiment, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

$\underline{\mathsf{Hybd}_1}$: It is the same as hybrid 0 except the challenger sets $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ and computes $\mathsf{CT}_0, \widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0), \sigma_{\mathsf{CT}}^*$ as before.

$\underline{\mathsf{Hybd}_2}$: It is the same as Hybd_1 except the computation of $V^* = (v_{k,\beta}^*)_{k \in [2\ell_c], \beta \in \{0,1\}}$. Let $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}}^*)$ be the challenge ciphertext where $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m_0^*)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ and $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. Then, the challenger computes $v_{k,\beta}^*$ as follows:

$$v_{k,\beta}^* \leftarrow \begin{cases} \mathsf{PRG}(u_{k,\beta_k}^*) \text{ for } u_{k,\beta_k}^* \leftarrow \{0,1\}^\lambda, & \text{if } \beta = \beta_k \\ \{0,1\}^{2\lambda}, & \text{if } \beta = 1 - \beta_k \end{cases}$$

for all $k \in [2\ell_c]$. Note that, the challenger defines $C_0 \coloneqq C_0[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ and sets $\sigma_{\mathsf{CT}}^* \coloneqq (u_{k,\beta}^*)_{k \in [2\ell_c]}$ as in the previous hybrid.

$\underline{\mathsf{Hybd}_3}$: It is the same as hybrid 1 except the challenger computes $\widetilde{C}_0^{\mathsf{slot}} \leftarrow i\mathcal{O}(1^\lambda, C_0^{\mathsf{slot}})$ instead of $\widetilde{C}_0$ where the circuit $C_0^{\mathsf{slot}} = C_0^{\mathsf{slot}}[\mathsf{MPK}, \mathsf{SK}_0, \mathsf{SK}_1, V^*, 0]$ is defined in Figure 2. The other components of the challenge ciphertext, i.e., $\mathsf{CT}_0, \mathsf{CT}_1$ remain the same as in the previous hybrid.

Constants: $\mathsf{MPK} = (\mathsf{hk}, h), \mathsf{SK}_0, \mathsf{SK}_1, V^* = (v^*_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}, j \in [0, L]$
Inputs: $\mathsf{sk}_i \in \{0,1\}^\lambda, i \in [L], \mathsf{pk}_i \in \{0,1\}^{2\lambda}, f_i \in \{0,1\}^{\ell_f}, \pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$, SKE ciphertexts $\{\mathsf{CT}_j\}_{j\in\{0,1\}}$ and $\sigma_{\mathsf{CT}} = (u_k)_{k\in[2\ell_c]}$

```
1. Parse $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$.
2. If $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{pk}_i, P_i), \pi_i) = 1 \wedge \mathsf{PRG}(\mathsf{sk}_i) = \mathsf{pk}_i \wedge (\mathsf{PRG}(u_k) = v^*_{k,\beta_k})_{k\in[2\ell_c]}$
3.    Compute $\widehat{m} \leftarrow \begin{cases} \mathsf{SKE.Dec}(\mathsf{SK}_0, \mathsf{CT}_0) \text{ if } i > j \\ \mathsf{SKE.Dec}(\mathsf{SK}_1, \mathsf{CT}_1) \text{ if } i \le j \end{cases}$
4.    Output $F(P_i, \widehat{m})$
5. Otherwise, output $\perp$
```

Fig. 2: The circuit $C^{\mathsf{slot}}_j = C^{\mathsf{slot}}_j[\mathsf{MPK}, \mathsf{SK}_0, \mathsf{SK}_1, V^*, j]$ for $j \in [0, L]$

$\underline{\mathsf{Hybd}_{3+j}}(j \in [L])$: It is the same as hybrid $2 + (j - 1)$ except the challenger computes $\widetilde{C}^{\mathsf{slot}}_j \leftarrow i\mathcal{O}(1^\lambda, C^{\mathsf{slot}}_j)$ instead of $\widetilde{C}^{\mathsf{slot}}_{j-1}$ where the circuit $C^{\mathsf{slot}}_j = C^{\mathsf{slot}}_j[\mathsf{MPK}, \mathsf{SK}_0, \mathsf{SK}_1, F, j]$ is defined in Figure 2.

$\underline{\mathsf{Hybd}_{4+L}}$: It is the same as hybrid $3 + L$ except the challenger computes $\widetilde{C}_1 \leftarrow i\mathcal{O}(1^\lambda, C_1)$ instead of $\widetilde{C}^{\mathsf{slot}}_L$ where the circuit $C_1 = C_1[\mathsf{MPK}, \mathsf{SK}_1, V^*]$ is defined in Figure 1. That is, the challenge ciphertext becomes $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_1, \sigma^*_{\mathsf{CT}})$.

$\underline{\mathsf{Hybd}_{5+L}}$: It is the same as hybrid $4 + L$ except the challenger sets $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m^*_1)$ and computes $\mathsf{CT}_1, \widetilde{C}_1 \leftarrow i\mathcal{O}(1^\lambda, C_0), \sigma^*_{\mathsf{CT}}$ as before.

$\underline{\mathsf{Hybd}_{6+L}}$: It is the same as hybrid $5 + L$ except the challenger computes $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$ instead of $\widetilde{C}_1$ where the circuit $C_0 = C_0[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ is defined in Figure 1.

$\underline{\mathsf{Hybd}_{7+L}}$: It is the same as Hybd_{6+L} except the computation of $V^* = (v^*_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}$. Let $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ be the challenge ciphertext where $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m^*_1)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m^*_1)$ and $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. Then, the challenger computes $v^*_{k,\beta}$ as follows:

$$v^*_{k,\beta} \leftarrow \begin{cases} \mathsf{PRG}(u^*_{k,\beta_k}) & \text{for } u^*_{k,\beta_k} \leftarrow \{0,1\}^\lambda, \text{ if } \beta = \beta_k \\ \mathsf{PRG}(u^*_{k,1-\beta_k}) & \text{for } u^*_{k,1-\beta_k} \leftarrow \{0,1\}^\lambda, \text{ if } \beta = 1 - \beta_k \end{cases}$$

for all $k \in [2\ell_c]$. Note that, the challenger defines $C_0 \coloneqq C_0[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ and sets $\sigma^*_{\mathsf{CT}} \coloneqq (u^*_{k,\beta})_{k\in[2\ell_c]}$ as in the previous hybrid.
$\underline{\mathsf{Hybd}_{8+L}}$: It is the same as hybrid $7 + L$ except the challenger sets $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m^*_1|})$. Observe that this hybrid is the same as $\mathsf{Expt}^{\mathsf{SlotRFE}}_{\mathcal{A}}(1^\lambda, 1)$.

Let $\mathsf{Hybd}^{\mathcal{A}}_i(\lambda)$ be the output of the hybrid experiment i. We show that the each pair of consecutive hybrids are indistinguishable from $\mathcal{A}$'s view in the following lemmas.

Lemma 1 *If* SKE *is IND-CPA secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}_0^{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Hybd}_1^{\mathcal{A}}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between hybrid 0 and 1 is that the challenge ciphertext component CT_1 is an SKE encryption of m_1^* instead of $\mathbf{0}_{|m_0^*|}$. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ who breaks the IND-CPA security of SKE with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives the slot count L from $\mathcal{A}$ and then plays the role of the challenger as in the hybrid 0 for the *setup* and *pre-challenge query* phase.
2. When $\mathcal{B}$ receives the challenge query $(\{(c_i, f_i, \mathsf{pk}_i^*)\}_{i\in[L]}, m_0^*, m_1^*)$ from $\mathcal{A}$, it works exactly the same as the challenger in hybrid 0 except it uses the SKE-challenger to compute CT_1. In particular, $\mathcal{B}$ sends the challenge message pair $(\mathbf{0}_{|m_0^*|}, m_1^*)$ to the SKE-challenger and gets back a ciphertext CT_1^*. Finally, $\mathcal{B}$ sends the challenge ciphertext $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1 \coloneqq \mathsf{CT}_1^*, \widetilde{C}_0, \sigma_{\mathsf{CT}}^*)$ to $\mathcal{A}$. Note that, $\mathcal{B}$ does not require the secret key SK_1 to compute the components CT_0, $\widetilde{C}_0, \sigma_{\mathsf{CT}}^*$ of ct^*.
3. At the end of the experiment, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the SKE-challenger computes $\mathsf{CT}_1^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m_0^*|})$ then $\mathcal{B}$ perfectly simulates hybrid 0. On the other hand, if the SKE-challenger computes $\mathsf{CT}_1^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ then $\mathcal{B}$ perfectly simulates hybrid 1. Therefore, $\mathcal{B}$ breaks the IND-CPA security of SKE with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the hybrids advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 2 *If* PRG *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}_1^{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Hybd}_2^{\mathcal{A}}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. We prove the lemma using a sequence of $2\ell_c$ hybrids $\mathsf{Hybd}_{1,k}$ for $k \in [2\ell_c + 1]$ where we sample $v_{t,1-\beta_t}^* \leftarrow \{0,1\}^{2\lambda}$ for all $t < k$ in $\mathsf{Hybd}_{1,k}$ and β_t represents the t-th bit of $(\mathsf{CT}_0, \mathsf{CT}_1)$. Note that, $\mathsf{Hybd}_{1,1}$ is identical to Hybd_1 and $\mathsf{Hybd}_{1,2\ell_c+1}$ is identical to Hybd_2. We only show that the distinguishing advantage of $\mathcal{A}$ between the hybrids $\mathsf{Hybd}_{1,k}$ and $\mathsf{Hybd}_{1,k+1}$ is negligible for each $k \in [2\ell_c]$. In particular, we show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ that breaks the security of PRG with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives a string $v^* \in \{0,1\}^{2\lambda}$ from the PRG-challenger.
2. $\mathcal{B}$ plays the role of the challenger as in the experiment $\mathsf{Hybd}_{1,k}$ or $\mathsf{Hybd}_{1,k+1}$. It receives the slot count L from $\mathcal{A}$ and runs the *setup* phase and sends crs to $\mathcal{A}$.
3. After that, $\mathcal{B}$ simulates the *pre-challenge query* phases as in $\mathsf{Hybd}_{1,k}$ or $\mathsf{Hybd}_{1,k+1}$.
4. At the *challenge query* phase, $\mathcal{B}$ computes $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m_0^*)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ and writes $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. Then, it computes $v_{t,\beta}^*$ as follows:

$$v_{t,\beta}^* \leftarrow \begin{cases} \mathsf{PRG}(u_{t,\beta_t}^*) \text{ for } u_{t,\beta_t}^* \leftarrow \{0,1\}^\lambda, & \text{if } \beta = \beta_t \\ \{0,1\}^{2\lambda}, & \text{if } \beta = 1-\beta_t \text{ and } t < k \\ v^* & \text{if } \beta = 1 - \beta_k \\ \mathsf{PRG}(u_{t,1-\beta_t}^*) \text{ for } u_{t,1-\beta_t}^* \leftarrow \{0,1\}^\lambda, & \text{if } \beta = 1-\beta_t \text{ and } t > k \end{cases}$$

for all $t \in [2\ell_c]$. Finally, it sets $V^* = (v^*_{t,\beta})_{t\in[2\ell_c],\beta\in\{0,1\}}$, $\sigma^*_{\mathsf{CT}} = (u^*_{t,\beta_t})_{t\in[2\ell_c]}$, $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$ where $C_0 = C_0[\mathsf{MPK}, \mathsf{sk}_j, V]$ and sends $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ to $\mathcal{A}$ as in the previous hybrid.
5. At the end of the experiment, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the PRG-challenger computes $v^* \leftarrow \mathsf{PRG}(u_{k,1-\beta_k})$ for some $u_{k,1-\beta_k} \leftarrow \{0,1\}^\lambda$ then $\mathcal{B}$ perfectly simulates $\mathsf{Hybd}_{1,k}$. On the other hand, if the PRG-challenger samples $v^* \leftarrow \{0,1\}^{2\lambda}$ then $\mathcal{B}$ perfectly simulates $\mathsf{Hybd}_{1,k+1}$. Therefore, $\mathcal{B}$ breaks the security of SSB with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the hybrids advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 3 *If $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}_2^{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Hybd}_3^{\mathcal{A}}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the hybrids 2 and 3 is that the ciphertext component $\widetilde{C}_0$ is replaced by $\widetilde{C}_0^{\mathsf{slot}}$. Since $i\mathcal{O}$ is secure it is sufficient to show that the two circuits C_0 and C_0^{slot} are equivalent. Let $(\mathsf{sk}_i, i, \mathsf{pk}_i, f_i, \pi_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$ be an arbitrary input to the circuits. The programming of the circuits differ only in *step* 2 where $\widehat{m}$ is computed via SKE decryption algorithm. The circuit C_0 always decrypts CT_0 using SK_0 whereas the circuit C_0^{slot} decrypts CT_0 using SK_0 if $i > 0$, otherwise it CT_1 using SK_1. Since $i \in [L]$ and $i > 0$ holds for all possible inputs $(\mathsf{sk}_i, i, \mathsf{pk}_i, f_i, \pi_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$, the circuit C_0^{slot} always decrypts CT_0 using SK_0 in step 2. Thus, the circuits C_0 and C_0^{slot} are equivalent. By the security of $i\mathcal{O}$, the distinguishing advantage of $\mathcal{A}$ is negligible in λ. □

Lemma 4 *If the* PRG *is secure,* SSB *is correct and secure, and $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}_{3+j}^{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Hybd}_{3+(j-1)}^{\mathcal{A}}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. We first introduce a new set of intermediate hybrids jHybd_{3+j} defined as follows:

$\underline{\mathsf{jHybd}_{3+j}}$: It works exactly the same as Hybd_{3+j} except the challenger samples $\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j+1)$ in the setup phase. The hash key hk binds with index $(j+1)$ instead of 1.

We now show that the distinguishing advantage of $\mathcal{A}$ between Hybd_{3+j} and jHybd_{3+j} is negligible for each $j \in [L]$.

Claim 1 If SSB satisfies index hiding then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{Hybd}_{3+j}^{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{jHybd}_{3+j}^{\mathcal{A}}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ who breaks the index hiding security of SSB with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives the slot count L from $\mathcal{A}$. Then, it sends L and $(1, j+1)$ to the SSB-challenger.
2. $\mathcal{B}$ receives a hash key hk^* from it's challenger and sets $\mathsf{hk} \coloneqq \mathsf{hk}^*$. Then, it sends $\mathsf{crs} \coloneqq \mathsf{hk}$ to $\mathcal{A}$.

3. After that, $\mathcal{B}$ plays the role of the challenger exactly similar to Hybd_{3+j} for simulating the *pre-challenge query* and *challenge* phases.
4. At the end of the experiment, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the SSB-challenger computes $\mathsf{hk}^* \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, 1)$ then $\mathcal{B}$ perfectly simulates Hybd_{3+j}. On the other hand, if the SSB-challenger computes $\mathsf{hk}^* \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j+1)$ then $\mathcal{B}$ perfectly simulates jHybd_{3+j}. Therefore, $\mathcal{B}$ breaks the index hiding security of SSB with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the hybrids advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

By Claim 1, proving Lemma 4 is equivalent ot prove the following claim.

Claim 2 If SSB is somewhere statistically binding then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1] - \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between $\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda)$ and $\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda)$ is in the second last component of the challenge ciphertext where it is $\widetilde{C}^{\mathsf{slot}}_j$ in hybrid $\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda)$. The circuits C^{slot}_j and C^{slot}_{j-1} behaves differently only for an input of the form $(\mathsf{sk}_j, j, \mathsf{pk}_j, f_j, \pi_j, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$. The analysis of the claim depends on whether the j-th user is corrupted or not. Let $(c_j, f_j, \mathsf{pk}^*_j)$ be the tuple specified by $\mathcal{A}$ during the challenge query phase. We define an event NonCorrupt as follows:

1. The index c_j satisfies $\{1, \ldots, \mathsf{ctr}\}$ meaning that pk_j was generated by the challenge on the c_j-th key generation query.
2. $\mathcal{A}$ never make a corruption query on index c_j.

We also denote $\overline{\mathsf{NonCorrupt}}$ by the event which is complement of NonCorrupt. By definition, we can write

$$\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1] = \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] + \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}]$$
$$\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1] = \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] + \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}]$$

Thus, it is sufficient to show that

$$|\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] - \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \mathsf{NonCorrupt}]| = \mathsf{negl}(\lambda) \quad (1)$$
$$|\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}] - \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}]| = \mathsf{negl}(\lambda) \quad (2)$$

We show that the equations 1 and 2 hold in claims 3 and 4 respectively.

Claim 3 If the PRG is secure, SSB is correct and secure, and $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] - \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \mathsf{NonCorrupt}]| = \mathsf{negl}(\lambda).$$

Proof. The main intuition for proving the claim is that the adversary $\mathcal{A}$ does not have sk_j and hence the associated public key pk_j can be chosen uniformly at random depending on the security of PRG. Then, with the help of SSB and $i\mathcal{O}$ we show that it is possible to change the obfuscated circuit from $\widetilde{C}^{\mathsf{slot}}_{j-1}$to $\widetilde{C}^{\mathsf{slot}}_j$. More precisely, we use the following sequence of hybrids:

$\mathsf{ncHybd}_{3+(j-1),1}$: It is the same as $\mathsf{jHybd}_{3+(j-1)}$ except at the beginning of the experiment, the challenger samples $q \leftarrow [Q]$ where $Q = Q(\lambda)$ denotes the total number of key generation queries $\mathcal{A}$ makes during the query phase. Let pk_q be the public key sampled by the challenger on the q-th key query (if there is one). The challenger aborts with output 0 if either of the following events occurs:

– $\mathcal{A}$ sends the tuple $(c_j, f_j, \mathsf{pk}_j^*)$ for registering the j-th user during the challenge query phase where $c_j \neq q$.

– $\mathcal{A}$ makes a corruption query with a index q.

Otherwise, the experiment proceeds exactly similar to $\mathsf{jHybd}_{3+(j-1)}$.

$\mathsf{ncHybd}_{3+(j-1),2}$: It is the same as $\mathsf{Hybd}_{3+(j-1),1}$ except the challenger samples $\mathsf{pk}_q \leftarrow \{0,1\}^{2\lambda}$ during the q-th key generation query. In this hybrid, the challenger is not required to answer for a corruption query on index q since it immediately aborts with output 0 as soon as it gets such a query.

$\mathsf{ncHybd}_{3+(j-1),3}$: It is the same as $\mathsf{ncHybd}_{3+(j-1),2}$ except the challenger obfuscates the circuit C_j^{slot} instead of C_{j-1}^{slot} while computing the challenge ciphertext.

$\mathsf{ncHybd}_{3+(j-1),4}$: It is the same as $\mathsf{ncHybd}_{3+(j-1),3}$ except the challenger samples $\mathsf{sk}_q \leftarrow \{0,1\}^{\lambda}$ and computes $\mathsf{pk}_q \leftarrow \mathsf{PRG}(\mathsf{sk}_q)$ during the q-th key generation query.

$\mathsf{ncHybd}_{3+(j-1),5}$: It is the same as $\mathsf{ncHybd}_{3+(j-1),4}$ except the challenger ignores the *abort condition* as defined in $\mathsf{ncHybd}_{3+(j-1),1}$.

As before, we denote $\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),k}$ by the output of the experiment $\mathsf{ncHybd}_{3+(j-1),k}$ for each $k \in [5]$. Next, we show the indistinguishability between any two consecutive hybrids in the following lemmas.

Lemma 5 *For all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, and $j \in [L]$ there exists a negligible function* negl *such that*

$$\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] = Q \cdot \Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),1} = 1].$$

Proof. By definition, the hybrids $\mathsf{jHybd}_{3+(j-1)}$ and $\mathsf{ncHybd}_{3+(j-1),1}$ proceeds exactly in the same way except the challenger aborts with output 0 if $c_j = q$ or $\mathcal{A}$ makes a corruption query for the index q. This means that both the experiments output 1 with the same probability if the event $\mathsf{NonCorrupt}$ occurs and $c_j = q$ holds. Thus, we can write

$$\begin{aligned}
&\Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),1} = 1] \\
&= \Pr[\mathsf{jHybd}_{3+(j-1)} = 1 \wedge \mathsf{NonCorrupt} \wedge c_j = q] \\
&= \Pr[c_j = q \mid \mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] \cdot \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] \\
&= 1/Q \cdot \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \mathsf{NonCorrupt}]
\end{aligned}$$

since the probability that $c_j = q$ holds where $q \leftarrow [Q]$ and $c_j \in \{1, \ldots, \mathsf{ctr}\} \subseteq [Q]$ is $1/Q$ given the event $\mathsf{NonCorrupt}$ has occurred. □

Lemma 6 *If* PRG *is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, and $j \in [L]$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),1} = 1] - \Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),2} = 1]| = \mathsf{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ that breaks the security of PRG with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ starts by sampling $q \leftarrow [Q]$ and receives a string $\mathsf{pk}^* \in \{0,1\}^{2\lambda}$ from the PRG-challenger.
2. $\mathcal{B}$ plays the role of the challenger as in the experiment $\mathsf{ncHybd}_{3+(j-1),1}$ or $\mathsf{ncHybd}_{3+(j-1),2}$. It receives the slot count L from $\mathcal{A}$ and runs the *setup* phase and sends crs to $\mathcal{A}$.
3. After that, $\mathcal{B}$ simulates the *pre-challenge query* phases as in $\mathsf{ncHybd}_{3+(j-1),1}$ or $\mathsf{ncHybd}_{3+(j-1),2}$. $\mathcal{B}$ returns pk^* when it receives a key generation query for the index q and adds $[\mathsf{ctr}] \mapsto (q, \mathsf{pk}^*, \perp)$ to D. If $\mathcal{A}$ makes a corruption query for the index q then $\mathcal{B}$ aborts with output 0.
4. At the *challenge query* phase, $\mathcal{B}$ checks if the tuple $(c_j, f_j, \mathsf{pk}_j^*)$ received from $\mathcal{A}$ satisfies $c_j = q$ and then it proceeds with the role of the challenger. If not, $\mathcal{B}$ aborts with output 0 as in $\mathsf{ncHybd}_{3+(j-1),1}$ or $\mathsf{ncHybd}_{3+(j-1),2}$.
5. At the end of the experiment, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the PRG-challenger computes $\mathsf{pk}^* \leftarrow \mathsf{PRG}(s)$ for some $s \leftarrow \{0,1\}^\lambda$ then $\mathcal{B}$ perfectly simulates $\mathsf{ncHybd}_{3+(j-1),1}$. On the other hand, if the PRG-challenger samples $\mathsf{pk}^* \leftarrow \{0,1\}^{2\lambda}$ then $\mathcal{B}$ perfectly simulates $\mathsf{ncHybd}_{3+(j-1),2}$. Therefore, $\mathcal{B}$ breaks the security of SSB with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the hybrids advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 7 *If* SSB *is somewhere statistically binding and* $i\mathcal{O}$ *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),2} = 1] - \Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),3} = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the hybrids is in the circuit which the challenger obfuscated during the challenge query phase: C^{slot}_{j-1} in $\mathsf{ncHybd}_{3+(j-1),2}$ and C^{slot}_{j} in $\mathsf{ncHybd}_{3+(j-1),3}$. We show that with overwhelming probability over the choice of hk and pk_q the circuits C^{slot}_{j} and C^{slot}_{j-1} are equivalent. Let us consider an arbitrary input $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}'_0, \mathsf{CT}'_1, \sigma_{\mathsf{CT}'})$ to the circuits. Note that the programming of the two circuits is different only in *step 2* (see Figure 2) where SKE decryption algorithm is performed. We consider the following cases:

Case 1: If $x \neq j$, then both the circuits either decrypt CT_0 when $x > j$ or CT_1 when $x < j$ in *step 2*. Hence, output of both the circuits is the same.

Case 2: If $x = j$ and $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_q, f_q)$, then we use the somewhere statistically binding property of SSB to argue that both the circuits return $\perp$. Note that, the challenger hardwires $\mathsf{MPK} = (\mathsf{hk}, h)$ in both the circuits computed as

$$\begin{aligned} \mathsf{hk} \leftarrow & \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j) \\ h \leftarrow & \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_q, f_q), \ldots (\mathsf{pk}_L, f_L)) \end{aligned}$$

in $\mathsf{ncHybd}_{3+(j-1),2}$ or $\mathsf{ncHybd}_{3+(j-1),3}$. By the somewhere statistically binding property of SSB, with overwhelming probability over the choice of hk (which binds index j), there does not exist any $(\mathsf{pk}^*, f^*) \neq (\mathsf{pk}_q, f_q)$ and π^* such that $\mathsf{SSB.Vrfy}(\mathsf{hk}, j, (\mathsf{pk}^*, f^*), \pi^*) = 1$. Therefore, if $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_q, f_q)$ then the circuits C^{slot}_{j} and C^{slot}_{j-1} output $\perp$ due to *step 1*.

Case 3: If $x = j$ and $(\mathsf{pk}_x, f_x) = (\mathsf{pk}_q, f_q)$, then the we use the fact that pk_q *is uniformly chosen* to argue that the circuits returns the same value. Let us assume the challenger does not abort in both experiments $\mathsf{ncHybd}_{3+(j-1),2}$, $\mathsf{ncHybd}_{3+(j-1),3}$. This means that $\mathsf{pk}_x = \mathsf{pk}_j = \mathsf{pk}_q$ where

$\mathsf{pk}_q \leftarrow \{0,1\}^{2\lambda}$ is the q-th public key. Since pk_q is chosen uniformly at random from $\{0,1\}^{2\lambda}$ then the probability that there exists some $s \in \{0,1\}^{\lambda}$ such that $\mathsf{PRG}(s) = \mathsf{pk}_q$ is at most $1/2^{\lambda}$ which is negligible in the security parameter. Therefore, with overwhelming probability it holds that $\mathsf{PRG}(\mathsf{sk}_x) \neq \mathsf{pk}_x = \mathsf{pk}_q$. Consequently, the check in *step 2* of both the circuits does not pass and as a result the circuits C_j^{slot} and C_{j-1}^{slot} output $\perp$.

Hence, for all possible inputs, the circuits C_j^{slot} and C_{j-1}^{slot} output the same value with overwhelming probability over the choice of $\mathsf{hk}, \mathsf{pk}_q$. Therefore, by the security of $i\mathcal{O}$, the lemma follows. ☐

Lemma 8 *If* PRG *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),3} = 1] - \Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),4} = 1]| = \mathsf{negl}(\lambda).$$

The proof follows similar to that of Lemma 6.

Lemma 9 *For all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \mathsf{NonCorrupt}] = Q \cdot \Pr[\mathsf{ncHybd}^{\mathcal{A}}_{3+(j-1),1} = 1].$$

The proof follows similar to that of Lemma 5.
Finally, the proof of Claim 3 follows by combining the Lemmas 5 to 9. ☐

Claim 4 If SSB is correct and secure, and $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}] - \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}]| = \mathsf{negl}(\lambda).$$

Proof. We prove the claim using the following two hybrid experiments:

$\underline{\mathsf{cHybd}_{3+(j-1),1}}$**:** It is the same as $\mathsf{jHybd}_{3+(j-1)}$ except the challenger aborts with output 0 if the event $\mathsf{NonCorrupt}$ occurs. This means that the output of the experiment can be 1 only if the public key pk_j is either adversarially generated or $\mathcal{A}$ makes a corruption query for the index j. Since $\mathcal{A}$ is admissible, in either cases, it must hold that $f_j(m_0^*) = f_j(m_1^*)$ where f_j is the associated function with pk_j.

$\underline{\mathsf{cHybd}_{3+(j-1),2}}$**:** It is the same as $\mathsf{cHybd}_{3+(j-1),1}$ except that the challenger obfuscates the circuit C_j^{slot} instead of C_{j-1}^{slot} while computing the challenge ciphertext.

As before, we denote $\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),k}$ by the output of the experiment $\mathsf{ncHybd}_{3+(j-1),k}$ for each $k \in \{1,2\}$. By definition, we have that

$$\begin{aligned}\Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+(j-1)}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}] &= \Pr[\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),1} = 1]\\ \Pr[\mathsf{jHybd}^{\mathcal{A}}_{3+j}(\lambda) = 1 \wedge \overline{\mathsf{NonCorrupt}}] &= \Pr[\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),2} = 1]\end{aligned}$$

Therefore, it is sufficient to prove that

$$|\Pr[\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),1} = 1] - \Pr[\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),2} = 1]| = \mathsf{negl}(\lambda).$$

We prove the indistinguishability between the hybrids in the following lemma.

Lemma 10 *If* SSB *is somewhere statistically binding and* $i\mathcal{O}$ *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),1} = 1] - \Pr[\mathsf{cHybd}^{\mathcal{A}}_{3+(j-1),2} = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the hybrids is in the circuit which the challenger obfuscated during the challenge query phase: C^{slot}_{j-1} in $\mathsf{cHybd}_{3+(j-1),1}$ and C^{slot}_{j} in $\mathsf{cHybd}_{3+(j-1),2}$. We show that with overwhelming probability over the choice of hk the circuits C^{slot}_{j} and C^{slot}_{j-1} are equivalent. Let us consider an arbitrary input $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}'_0, \mathsf{CT}'_1, \sigma_{\mathsf{CT}'})$ to the circuits. Note that the programming of the two circuits is different only in *step 2* (see Figure 2) where SKE decryption algorithm is performed. We consider the following cases:

Case 1: If $x \neq j$, then both the circuits either decrypt CT_0 when $x > j$ or CT_1 when $x < j$ in *step 2*. Hence, output of both the circuits is the same.

Case 2: If $x = j$ and $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_j, f_j)$, then we use the somewhere statistically binding property of SSB to argue that both the circuits return $\perp$. Note that, the challenger hardwires $\mathsf{MPK} = (\mathsf{hk}, h)$ in both the circuits computed as

$$\begin{aligned} \mathsf{hk} &\leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j) \\ h &\leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_j, f_j), \ldots (\mathsf{pk}_L, f_L)) \end{aligned}$$

in $\mathsf{cHybd}_{3+(j-1),1}$ or $\mathsf{cHybd}_{3+(j-1),2}$. By the somewhere statistically binding property of SSB, with overwhelming probability over the choice of hk (which binds index j), there does not exist any $(\mathsf{pk}^*, f^*) \neq (\mathsf{pk}_j, f_j)$ and π^* such that $\mathsf{SSB.Vrfy}(\mathsf{hk}, j, (\mathsf{pk}^*, f^*), \pi^*) = 1$. Therefore, if $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_j, f_j)$ then the circuits C^{slot}_{j} and C^{slot}_{j-1} output $\perp$ due to *step 1*.

Case 3: If $x = j$ and $(\mathsf{pk}_x, f_x) = (\mathsf{pk}_j, f_j) \wedge (\mathsf{CT}'_0 \neq \mathsf{CT}_0 \vee \mathsf{CT}'_1 \neq \mathsf{CT}_1)$, then we use the fact that $v^*_{k,1-\beta_k}$'s are chosen uniformly from $\{0,1\}^{2\lambda}$ for all $k \in [2\ell_c]$ to argue that both the circuits return $\perp$. Suppose $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c})$ and $(\mathsf{CT}'_0, \mathsf{CT}'_1) = (\beta'_1, \ldots, \beta'_{\ell_c}, \beta'_{\ell_c+1}, \ldots, \beta'_{2\ell_c})$. Since $\mathsf{CT}'_0 \neq \mathsf{CT}_0$ or $\mathsf{CT}'_1 \neq \mathsf{CT}_1$ there exists $t \in [2\ell_c]$ such that $\beta'_t = 1 - \beta_t$. Let us assume $\sigma_{\mathsf{CT}'} = (u'_k)_{k \in [2\ell_c]}$ where $u'_k \in \{0,1\}^\lambda$ for all $k \in [2\ell_c]$. In order to pass the check of *step 2* in both the circuits, it should hold that $\mathsf{PRG}(u'_t) = v^*_{t,\beta'_t}$. Since $v^*_{t,\beta'_t} = v^*_{t,1-\beta_t}$ is chosen uniformly at random from $\{0,1\}^{2\lambda}$ then the probability that there exists $u' \in \{0,1\}^\lambda$ such that $\mathsf{PRG}(u') = v^*_{t,1-\beta_t}$ is at most $1/2^\lambda$ which is negligible in the security parameter. Therefore, with overwhelming probability it holds that $\mathsf{PRG}(u'_t) \neq v^*_{t,\beta'_t}$. Consequently, the check in *step 2* of both the circuits does not pass and as a result the circuits C^{slot}_{j} and C^{slot}_{j-1} output $\perp$.

Case 4: If $x = j$ and $(\mathsf{pk}_x, f_x) = (\mathsf{pk}_j, f_j) \wedge (\mathsf{CT}'_0 = \mathsf{CT}_0 \wedge \mathsf{CT}'_1 = \mathsf{CT}_1)$, then we use the fact that pk_j is corrupted and $\mathcal{A}$ is admissible. Let us assume the challenger does not abort in both experiments $\mathsf{cHybd}_{3+(j-1),1}$, $\mathsf{cHybd}_{3+(j-1),2}$. This means that $\mathsf{pk}_x = \mathsf{pk}_j$ where pk_j is either adversarially generated or it is corrupted. Assuming that the check of *step 2* passes in both the circuits, the SKE decryption algorithm of *step 2* recovers m^*_0 from CT_0 in C^{slot}_{j-1} whereas it recovers m^*_1 from CT_1 in C^{slot}_{j}. Consequently, on input $(\mathsf{sk}_j, j, \mathsf{pk}_j, f_j, \pi_j, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$ the circuit C^{slot}_{j-1} outputs $f_j(m^*_0)$ and the circuit C^{slot}_{j} outputs $f_j(m^*_1)$. Since $\mathcal{A}$ is admissible, we have $f_j(m^*_0) = f_j(m^*_1)$. In other words, both the circuits C^{slot}_{j} and C^{slot}_{j-1} output the same value.

Hence, for all possible inputs, the circuits C^{slot}_{j} and C^{slot}_{j-1} output the same value with overwhelming probability over the choice of $\mathsf{hk}, \mathsf{pk}_q$. Therefore, by the security of $i\mathcal{O}$, the lemma follows. □

Combining Lemma 2 and 10, the Claim 4 holds. $\square$

Therefore, the proof of Claim 2 follows from Claims 3 and 3. $\square$

Finally, the proof of Lemma 4 follows from the Claim 2. $\square$

Lemma 11 *If $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}^{\mathcal{A}}_{3+L}(\lambda) = 1] - \Pr[\mathsf{Hybd}^{\mathcal{A}}_{4+L}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 11 follows from a similar argument as in Lemma 3.

Lemma 12 *If* SKE *is IND-CPA secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}^{\mathcal{A}}_{4+L}(\lambda) = 1] - \Pr[\mathsf{Hybd}^{\mathcal{A}}_{5+L}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between hybrid $4 + L$ and $5 + L$ is that the challenge ciphertext component CT_0 is an SKE encryption of m_1^* instead of m_0^*. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ who breaks the IND-CPA security of SKE with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives the slot count L from $\mathcal{A}$ and then plays the role of the challenger as in the hybrid $3 + L$ for the *setup* and *pre-challenge query* phase.
2. When $\mathcal{B}$ receives the challenge query $(\{(c_i, f_i, \mathsf{pk}_i^*)\}_{i\in[L]}, m_0^*, m_1^*)$ from $\mathcal{A}$, it works exactly the same as the challenger in hybrid $3 + L$ except it uses the SKE-challenger to compute CT_0. In particular, $\mathcal{B}$ sends the challenge message pair (m_0^*, m_1^*) to the SKE-challenger and gets back a ciphertext CT_0^*. Finally, $\mathcal{B}$ sends the challenge ciphertext $\mathsf{ct}^* = (\mathsf{CT}_0 \coloneqq \mathsf{CT}_0^*, \mathsf{CT}_1, \widetilde{C}_1, \sigma^*_{\mathsf{CT}})$ to $\mathcal{A}$. Note that, $\mathcal{B}$ does not require the secret key SK_0 to compute the components CT_1, $\widetilde{C}_1, \sigma^*_{\mathsf{CT}}$ of ct^*.
3. At the end of the experiment, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ which is also the output of $\mathcal{B}$.

If the SKE-challenger computes $\mathsf{CT}_0^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_0^*)$ then $\mathcal{B}$ perfectly simulates hybrid $4+L$. On the other hand, if the SKE-challenger computes $\mathsf{CT}_0^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ then $\mathcal{B}$ perfectly simulates hybrid $5 + L$. Therefore, $\mathcal{B}$ breaks the IND-CPA security of SKE with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the hybrids advantage $\epsilon(\lambda)$. Hence, the lemma follows. $\square$

Lemma 13 *If $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}^{\mathcal{A}}_{5+L}(\lambda) = 1] - \Pr[\mathsf{Hybd}^{\mathcal{A}}_{6+L}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the hybrids $5+L$ and $6+L$ is that the ciphertext component $\widetilde{C}_1$ is replaced by $\widetilde{C}_1$. Since $i\mathcal{O}$ is secure it is sufficient to show that the two circuits C_0 and C_1 are equivalent. Let $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}'_0, \mathsf{CT}'_1, \sigma'_{\mathsf{CT}})$ be an arbitrary input to the circuits. The programming of the circuits differ only in *step 3* where $\widehat{m}$ is computed via SKE decryption algorithm. The circuit C_0 always decrypts CT_0 using SK_0 whereas the circuit C_1 decrypts CT_1 using SK_1.

Let $(\mathsf{CT}_0, \mathsf{CT}_1)$ be the part of the challenge ciphertext of hybrid $5 + L$ or $6 + L$. Suppose $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c})$ and $(\mathsf{CT}'_0, \mathsf{CT}'_1) = (\beta'_1, \ldots, \beta'_{\ell_c}, \beta'_{\ell_c+1}, \ldots, \beta'_{2\ell_c})$. We have two cases:

Case 1: If $(\mathsf{CT}'_0, \mathsf{CT}'_1) = (\mathsf{CT}_0, \mathsf{CT}_1)$, then both the circuits compute $m_1^* \leftarrow \mathsf{SKE.Dec}(\mathsf{SK}_j, \mathsf{CT}_j)$ for $j \in \{0,1\}$ (assuming that the check of *step 2* passes for both the circuits). Therefore, output of the circuits C_0 and C_1 are the same for an input of the from $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}_0, \mathsf{CT}_1, \sigma'_{\mathsf{CT}})$.

Case 2: If $(\mathsf{CT}'_0, \mathsf{CT}'_1) \neq (\mathsf{CT}_0, \mathsf{CT}_1)$, then we rely on the formation of V^* to argue that the circuits C_0 and C_1 output $\perp$. Since $\mathsf{CT}'_0 \neq \mathsf{CT}_0$ or $\mathsf{CT}'_1 \neq \mathsf{CT}_1$ there exists $t \in [2\ell_c]$ such that $\beta'_t = 1 - \beta_t$. Let us assume $\sigma_{\mathsf{CT}'} = (u'_k)_{k \in [2\ell_c]}$ where $u'_k \in \{0,1\}^\lambda$ for all $k \in [2\ell_c]$. In order to pass the check of *step 2* in both the circuits, it should hold that $\mathsf{PRG}(u'_t) = v^*_{t,\beta'_t}$. Since $v^*_{t,\beta'_t} = v^*_{t,1-\beta_t}$ is chosen uniformly at random from $\{0,1\}^{2\lambda}$ then the probability that there exists $u' \in \{0,1\}^\lambda$ such that $\mathsf{PRG}(u') = v^*_{t,1-\beta_t}$ is at most $1/2^\lambda$ which is negligible in the security parameter. Therefore, with overwhelming probability it holds that $\mathsf{PRG}(u'_t) \neq v^*_{t,\beta'_t}$. Consequently, the check in *step 2* of both the circuits does not pass and as a result the circuits C_0 and C_1 output $\perp$.

Thus, the circuits C_0 and C_1 are equivalent over the choice of V^*. By the security of $i\mathcal{O}$, the distinguishing advantage of $\mathcal{A}$ is negligible in λ. □

Lemma 14 *If* PRG *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}^{\mathcal{A}}_{6+L}(\lambda) = 1] - \Pr[\mathsf{Hybd}^{\mathcal{A}}_{7+L}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 14 follows from a similar argument as in Lemma 2.

Lemma 15 *If* SKE *is IND-CPA secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Hybd}^{\mathcal{A}}_{7+L}(\lambda) = 1] - \Pr[\mathsf{Hybd}^{\mathcal{A}}_{8+L}(\lambda) = 1]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 15 follows from a similar argument as in Lemma 1.

Finally, the proof the Theorem 1 follows from combining the proofs of the Lemmas 1 to 4 and Lemmas 11 to 15. □

6 From Slotted Registered FE to Registered FE

Hohenberger et al. [7] showed a transformation from slotted registered ABE to registered ABE. The same transformation also works for the case of registered FE. Roughly, they use a simple "powers-of-two" approach for the conversion. If we want to support $L = 2^\ell$ users in the system then the transformation utilizes $(\ell + 1)$ copies of slotted registered ABE to achieve a registered ABE. Fortunately, the same approach also works for the case of FE. We skip the details since it is identical to the Section 6 of [7].

References

1. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: CRYPTO 2001. pp. 213–229. Springer (2001)
2. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: TCC 2011. pp. 253–273. Springer (2011)
3. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)

4. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A.: Registration-based encryption: Removing private-key generator from IBE. In: TCC 2018. pp. 689–718. Springer (2018). https://doi.org/10.1007/978-3-030-03807-6_25
5. Gay, R., Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. IACR Cryptololy ePrint Archive, Report 2020/764 (2020)
6. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. Journal of the ACM JACM **62**(6), 1–33 (2015)
7. Hohenberger, S., Lu, G., Waters, B., Wu, D.J.: Registered attribute-based encryption. Cryptology ePrint Archive, Paper 2022/1500 (2022), https://eprint.iacr.org/2022/1500, https://eprint.iacr.org/2022/1500
8. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. arXiv preprint arXiv:2008.09317 (2020)
9. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: CRYPTO 2012. pp. 180–198. Springer (2012)
10. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive, Report 2010/556 (2010)
11. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT 2005. pp. 457–473. Springer (2005)
12. Shamir, A.: Identity-based cryptosystems and signature schemes. In: CRYPTO 1984. pp. 47–53. Springer (1984)