Trace and Revoke with Optimal Parameters from Polynomial Hardness

Shweta Agrawal[*] Simran Kumari[†] Anshu Yadav[‡] Shota Yamada[§]

Abstract

A *trace and revoke* (TR) scheme is an N user traitor tracing scheme which additionally enables the encryptor to specify a list $L \subseteq$ of revoked users so that these users can no longer decrypt ciphertexts. The "holy grail" of this line of work is a construction which resists unbounded collusions, achieves ciphertext, public and secret key sizes independent[1] of $|L|$ and $|N|$, and is based on polynomial hardness assumptions. In this work we make the following contributions:

1. *Public Trace Setting:* We provide a construction which (i) achieves optimal parameters, (ii) supports embedding identities (from an exponential space) in user secret keys, (iii) relies on polynomial hardness assumptions, namely compact functional encryption (FE) and a key-policy attribute based encryption (ABE) with special efficiency properties constructed by Boneh et al. (Eurocrypt 2014) from Learning With Errors (LWE), and (iv) enjoys adaptive security with respect to the revocation list. The previous best known construction by Nishimaki, Wichs and Zhandry (Eurocrypt 2016) which achieved optimal parameters and embedded identities, relied on indistinguishability obfuscation, which is considered an inherently subexponential assumption and achieved only selective security with respect to the revocation list.

2. *Secret Trace Setting:* We provide the first construction with optimal ciphertext, public and secret key sizes and embedded identities from any assumption outside Obfustopia. In detail, our construction relies on Lockable Obfuscation which can be constructed using LWE (Goyal, Koppula, Waters and Wichs, Zirdelis, Focs 2017) and two ABE schemes: (i) the key-policy scheme with special efficiency properties by Boneh et al. (Eurocrypt 2014) and (ii) a ciphertext-policy ABE for P which was recently constructed by Wee (Eurocrypt 2022) using a new assumption called *evasive and tensor* LWE. This assumption, introduced to build an ABE, is believed to be much weaker than lattice based assumptions underlying FE or iO – in particular it is required even for lattice based broadcast, without trace.

   Moreover, by relying on subexponential security of LWE, both our constructions can also support a *super-polynomial* sized revocation list, so long as it allows efficient representation and membership testing. Ours is the first work to achieve this, to the best of our knowledge.

[*]IIT Madras, shweta@cse.iitm.ac.in

[†]IIT Madras, sim78608@gmail.com

[‡]IIT Madras, anshu.yadav06@gmail.com

[§]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, yamada-shota@aist.go.jp

[1]Ignoring logarithmic dependencies.

Contents

1 Introduction

Traitor Tracing. Traitor tracing (TT) schemes were first proposed by Chor, Fiat, and Naor [CFN94] to enable content providers to trace malicious users who exploit their secret keys to construct illegal decryption boxes. More formally, a TT system is a public key encryption system comprising N users for some large polynomial N. Each user $i \in [N]$ is provided with a unique secret key sk_i for decryption, and there is a common public key pk which is used by the content distributor to encrypt content. If any collection of users attempts to create and sell a new decoding box that can be used to decrypt the content, then the tracing algorithm, given black-box access to any such pirate decoder, is guaranteed to output an index $i \in [N]$ of one of the corrupt users, which in turn allows to hold them accountable. The literature has considered both public and secret tracing, where the former requires knowledge of a secret key to run the trace procedure and the latter does not suffer from this restriction.

Broadcast Encryption. Broadcast Encryption [FN93] (BE) introduced by Fiat and Naor, is also an N user system which supports an encrypted broadcast functionality. In BE, a content provider can transmit a single ciphertext over a broadcast channel so that only an authorized subset $S \subseteq N$ of users can decrypt and recover the message. More formally, each user $i \in [N]$ is provided with a unique decryption key sk_i and a ciphertext ct_m for a message m also encodes an authorized list S so that sk_i decrypts ct_m if and only if $i \in S$. Evidently, public key encryption provides a trivial construction of BE with ciphertext of size $O(N)$ – thus, the focus in such schemes is to obtain short ciphertext, ideally logarithmic in N.

Trace and Revoke. Naor and Pinkas [NP10] suggested a meaningful interleaving of these two functionalities so that traitors that are identified by the TT scheme can be removed from the set of authorized users in a BE scheme. To capture this, they defined the notion of "Trace and Revoke" (TR) where the content provider in a broadcast encryption scheme includes a list L of revoked users in the ciphertext, and sk_i works to decrypt ct_L if $i \notin L$. Moreover, it is required that revocation remain compatible with tracing, so that if an adversary builds a pirate decoder that can decrypt ciphertexts encrypted with respect to L, then the tracing algorithm should be able to output a corrupt non-revoked user who participated in building the illegal decoder. Trace and revoke systems provide a functionality which is richer than a union of BE and TT, since the traitor traced by the latter must belong to the set of non-revoked users for the guarantee to be meaningful. As such, TT schemes have been challenging to construct even given TT and BE schemes.

The Quest for Optimal Parameters. All the above primitives have been researched extensively over decades, resulting in a long sequence of beautiful constructions, non-exhaustively [BGW05, BW06, BWZ14, BZ17, NWZ16, KW20, GKW18, GKW19, AY20, Wee22]. A central theme in this line of work is to achieve optimal parameters, namely optimal sizes for the ciphertext, public key and secret key (and understanding tradeoffs thereof), while still supporting unbounded collusion resistance. Towards this, the powerful hammer of indistinguishability obfuscation (iO) [BGI+01] yielded the first feasibility results for traitor tracing [BZ17] as well as trace and revoke [NWZ16] while multilinear maps [GGH13, CLT13] led to the first construction for broadcast encryption [BWZ14]. Though there has been remarkable progress in the construction of iO from standard assumptions, with the breakthrough work of Jain, Lin and Sahai [JLS21, JLS22] finally reaching this goal, iO is an inherently subexponential assumption [GPSZ17] because the challenger is required to check whether two circuits are functionally equivalent, which can take exponential time in general. Indeed, all known constructions of iO assume subexponential hardness of the underlying algebraic assumptions. To address this limitation, a sequence of works [GPSZ17, AM18, BKS16, GPS16, KS20] has sought to replace iO by polynomially hard assumptions such as functional encryption in different applications.

Optimal TT, BE and TR from Polynomial Assumptions: For traitor tracing, the first construction from standard assumptions was finally achieved by the seminal work of Goyal, Koppula and Waters [GKW18] in the secret trace setting, from the Learning With Errors (LWE) assumption. For broadcast encryption, this goal was achieved by Agrawal and Yamada [AY20] from LWE and the bilinear GGM. In the standard model, Agrawal, Wichs and Yamada [AWY20] provided a construction from a non-standard knowledge assumption on pairings, while Wee [Wee22] provided a construction from a new assumption on lattices, called Evasive and Tensor LWE. For trace and revoke, the only construction without iO that achieves collusion resistance and optimal parameters is by Goyal, Vusirikala and Waters (GVW) [GVW19] from *positional witness encryption* (PWE) which is a polynomial hardness assumption. However, their construction incurs an exponential loss in the security proof, requiring the underlying PWE to satisfy subexponential security. Moreover, although PWE is not an inherently subexponential assumption as are iO and witness encryption (WE), we do not currently know of any *constructions* of PWE that rely on standard polynomial hardness assumptions. In particular, [JLS21, JLS22] do *not* imply PWE from polynomial hardness.

Pathway via Secret Tracing. Both the iO and PWE based constructions of TR [NWZ16, GVW19] achieve public tracing. Taking a lesson from TT, where optimal parameters were achieved from standard assumptions only in the secret trace setting [GKW18], a natural approach towards optimal TR from better assumptions is to weaken the tracing algorithm to be secret key. This approach has been explored in a number of works – the current best parameters are achieved by Zhandry [Zha20] who obtains the best known tradeoff in ciphertext, public key and secret key size. In particular, Zhandry [Zha20] showed that all parameters can be of size $O(N^{1/3})$ by relying on the bilinear generic group model (GGM). Note that the generic group model is a strong assumption, and indeed a construction secure in this model cannot be considered as relying on standard assumptions, since several non-standard assumptions on pairings are secure in the GGM. Prior to [Zha20], Goyal et al. [GQWW19] provided a construction from LWE and Pairings, but their overall parameters are significantly worse – while their ciphertext can be arbitrarily small, $O(N^{\epsilon})$, their public key is $O(N)$ and secret key is $O(N^{c})$ for some large constant c[2].

Thus, a central open question in TR is:

Can we construct collusion resistant Trace and Revoke with optimal parameters from concrete polynomial assumptions?

Embedding Identities. Traditionally, it was assumed that tracing the index $i \in [N]$ of a corrupt user is enough, and there is an external mapping, maintained by the content distributor or some other party which associates the number i to the identity of the user, i.e. name, national identity number and such, which is then used to ensure accountability. The work of Nishimaki, Wichs and Zhandry (henceforth NWZ) [NWZ16] argued that this assumption is problematic since it implies that a user must trust the content provider with her confidential information. Storing such a map is particularly worrisome in the setting of public tracing since the user either cannot map the recovered index to an actual person, or the index-identity map must be stored publicly.

NWZ provided an appealing solution to the above conundrum – they suggest that identifying information be embedded in the key of the user, so that if a coalition of traitors constructs a pirate decoder, the tracing algorithm can directly retrieve the identifying information from

[2] Zhandry [Zha20] states that the secret size in [GQWW19] is $O(N^2)$ but in fact the exponent is much larger due to the usage of arithmetic computations in NC^1, which blows up the circuit size associated with the ABE secret keys.

one of the keys that was used to construct the decoder and no one needs to keep any records associating users to indices. Notably, the identities can live in an *exponential* sized space, which introduces significant challenges in the tracing procedure. Indeed, handling an exponential space in the tracing procedure is the key contribution of NWZ. They also provided constructions of traitor tracing as well as trace and revoke with embedded identities, denoted by EITT and EITR respectively, from various assumptions.

1.1 Prior Work: Embedded Identity Trace and Revoke.

In the *public* trace setting, the only work that achieves embedded identity trace and revoke (EITR) with full collusion resistance is that of NWZ. However, while it takes an important first step, the construction by NWZ suffers from the following drawbacks:

1. *Reliance on Subexponential Hardness Assumption.* The construction relies on indistinguishability obfuscation [BGI+01], which appears to be an inherently subexponential assumption as discussed above.

2. *Selective Security in Revocation List:* Despite relying on adaptive security of functional encryption, the notion of security achieved by their construction is selective – the adversary must announce the revocation list before making any key requests or seeing the challenge ciphertext.

In the *secret* trace setting, the work of Kim and Wu [KW20] achieves EITR from the subexponential Learning With Errors (LWE) assumption. However, their construction incurs a ciphertext size that grows with the size of the revocation list. Additionally, while they can achieve adaptive security with respect to the revocation list, this is either by incurring an exponential loss in the security proof, or by assuming sub-exponential security for an ingredient scheme.

1.2 Our Results

In this work, we provide the first constructions with optimal parameters from polynomial assumptions, which additionally support embedded identities from an exponential space. We detail our contributions below.

Public Trace Setting. We provide a construction of Trace and Revoke with public tracing which overcomes the limitations of NWZ – (i) it relies on polynomial hardness assumptions, namely functional encryption and "special" attribute based encryption, both of which can be constructed using standard polynomial hardness assumptions [BGG+14, JLS21, JLS22] (ii) it enjoys adaptive security in the revocation list.

A detailed comparison with prior work is provided in Table 1.

Our Assumptions. Functional Encryption (FE) and Attribute Based Encryption (ABE) are generalizations of Public Key Encryption. In FE, a secret key corresponds to a circuit C and a ciphertext corresponds to an input x from the domain of C. Given a function key sk_C and a ciphertext ct_x, the decryptor can learn $C(x)$ and nothing else. It has been shown that FE implies iO [AJ15, BV15] albeit with exponential loss. The aforementioned work of Jain, Lin and Sahai [JLS21, JLS22] provides a construction of compact FE from polynomial hardness assumptions, namely LPN, PRG in NC_0 and pairings. ABE is a special case of FE in which the input can be divided into a public and private part (x, m) and the circuit C in the secret key sk_C is only

Work	|CT|	|SK|	|PK|	Trace Space	Sel/Adp	Asspn	Identities
[NWZ16]	1	1	1	Exp	Selective	Subexp (iO)	Yes
[GVW19]	1	1	1	Poly	Adaptive	Subexp (subexp PWE)	No
This	1	1	1	Exp	Adaptive	Poly (FE and Special ABE)	Yes

Table 1: State of the art with Public Traceability.

evaluated on the public part x in the ciphertext $\mathsf{ct}_{x,m}$. The private message m is revealed by decryption if and only if $C(x) = 1$. While FE implies ABE in general, we require our underlying ABE to satisfy special efficiency properties, which is not generically implied by FE. However, the desired ABE can be instantiated using the construction of Boneh et al. [BGG+14] which is based on LWE.

Secret Trace Setting. In the secret trace setting, we achieve the optimal size of $O(\log N)$ for ciphertext, public and secret key by relying on Lockable Obfuscation (LO) [GKW17, WZ17] and two special ABE schemes – one, the key-policy scheme with special efficiency properties by Boneh et al. [BGG+14] which is based on LWE, and two, a ciphertext-policy ABE for P which was recently constructed by Wee [Wee22] using the new evasive and tensor LWE assumption. The new assumptions by Wee, also independently discovered by Tsabary [Tsa22], are formulated for designing ciphertext-policy ABE which is much weaker than FE since ABE is an all or nothing primitive in contrast to FE. As such, these are believed to be much weaker than lattice based assumptions that have been introduced in the context of FE or iO.

Thus, we can replace the use of FE in the public trace setting by another ABE (though of type ciphertext-policy) and an LO in the private trace setting, albeit via a completely different construction. Along the way, we show that a small modification to the TR construction by Goyal et al. (Crypto 2019) yields a ciphertext of size $O(\log N)$ as against their original $O(N^\epsilon)$, from LWE and pairings. However, this construction retains the large public and secret keys of their construction, which depend at least linearly on N. Our results are summarized in Table 2.

Super-polynomial Revoke List. Lastly, by relying on subexponential security of LWE, both our constructions can support a *super-polynomial* sized revocation list, so long as it allows efficient representation and membership testing. Ours is the first work to achieve this, to the best of our knowledge.

1.3 Technical Overview

We proceed to give an overview of our techniques. We begin by defining the notion of revocable predicate encryption (RPE) in both the public and secret setting, then describe the ideas used to instantiate this primitive. Finally we outline how to upgrade public/secret RPE to build trace and revoke with embedded identities with public/secret tracing.

Work	$\lvert\mathsf{CT}\rvert$	$\lvert\mathsf{SK}\rvert$	$\lvert\mathsf{PK}\rvert$	Trace Space	Asspn	Identities
[GQWW19]	N^{ϵ}	N^{poly}	N	Poly	LWE and Pairings	No
[Zha20]	N^{a}	N^{1-a}	N^{1-a}	Poly	GGM Pairings	No
[KW20]	L	1	1	Exp	Subexp LWE	Yes
This	1	1	1	Exp	Evasive Tensor LWE	Yes
Modified [GQWW19]	1	N^{c}	N	Poly	LWE and Pairings	No

Table 2: State of the Art with Secret Traceability. The column $\lvert\mathsf{CT}\rvert$ captures the dependence of ciphertext size on N and L where N denotes the number of users and L denotes the length of the revocation list. Parameters that are logarithmic in N, L or polynomial in the security parameter are represented as 1. Here, $0 < a < 1$ and $\epsilon > 0$ can be chosen arbitrarily. c is a large constant.

Revocable Predicate Encryption. NWZ introduced the notion of revocable functional encryption (RFE) and used it to construct EITR with public tracing. Subsequently, Kim and Wu [KW20] adapted this notion to the secret key setting, under the name of revocable predicate encryption (RPE) and used it to construct EITR with secret tracing. In this work, we extend Kim and Wu's notion of RPE to the public key setting and use it to construct EITR with public tracing. Our notion of RPE in the public setting is similar to but weaker than RFE[3] – it only supports "all or nothing" decryption in contrast to RFE. This weaker notion nevertheless suffices to construct EITR and moreover admits constructions from weaker assumptions.

In RPE, the key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}$ and an attribute $x \in \mathcal{X}$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},x}$. The encryption algorithm takes as input the encryption key ek, a function f, a message $m \in \mathcal{M}$, and a revocation list $L \subseteq \mathcal{L}$. It outputs a ciphertext ct. Decryption recovers m if $f(x) = 1$ and $\mathsf{lb} \notin L$. In the public variant of RPE, ek is a public key, while in the secret variant, ek is a secret key. In the secret variant, the scheme is also required to support a public "broadcast" functionality, i.e. there exists a public encryption algorithm that allows anyone to encrypt a message with respect to the "always-accept" policy, i.e. a policy that evaluates to true for all inputs. This is analogous to the primitive of "mixed FE" introduced by [GKW18].

In terms of security, we require RPE to satisfy message hiding and function hiding. At a high level, message hiding stipulates that an adversary cannot distinguish between encryptions of (f, m_0) and (f, m_1) as long as every key query for (lb, x) satisfies $f(x) = 0$ or $\mathsf{lb} \in L$. Function hiding stipulates that an adversary cannot distinguish between encryptions of (f_0, m) and (f_1, m) as long as every key query for (lb, x) satisfies $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

Before we describe our constructions, we highlight the chief difficulties that are inherent to designing RPE:

1. *Independence of parameter sizes from $|L|$.* A key requirement in TR schemes is that the ciphertext size should be independent of the length of the revocation list L – this constraint

[3]Syntactically, RPE is "ciphertext-policy" while RFE is "key-policy", i.e. the function is emdedded in the ciphertext in RPE as against the key in RFE.

must also be satisfied by the underlying RPE, in both the secret and public setting. In our work, we insist that even the public and secret keys satisfy $|L|$ independence. This constraint is inherited from broadcast encryption, and is challenging to satisfy. Further, note that L must be unbounded – its length cannot be fixed during setup, which introduces additional difficulties.

2. *Encrypted Computation.* While the revocation list L need not be hidden by the ciphertext, the function f[4] in the ciphertext is required to be hidden, as formalized by our function hiding requirement. Yet, this hidden function must participate in computing $f(x)$ where x is provided in the key. This requirement makes TR schemes worryingly close to collusion resistant functional encryption, an "obfustopia" primitive which we want to avoid in the secret trace setting.

Constructing Public Revocable Predicate Encryption. We proceed to describe the main ideas in constructing public RPE.

Overview of NWZ. The work of NWZ addresses the challenge of making the ciphertext size independent of $|L|$ by using a somewhere statistically binding (SSB) hash and hides the function f by using a functional encryption scheme, where f is encrypted in the ciphertext. However, they must additionally rely on iO – at a high level, this is because they require the *decryptor* to compute the SSB opening π and then run SSB verification on it (details of how SSB algorithms work are not relevant for this overview). In turn, the reason they need the decryptor to compute the opening π is because this needs both the set L and the label lb, which are available only to the decryptor – note that the encryptor has only L and the key generator has only lb. Now, since the decryptor has to compute π and run SSB verification, and since the program that computes SSB verification has some secrets, the decryptor is allowed to obtain obfuscation of this program. To implement this idea, they nest iO inside a compact FE scheme so that FE decryption outputs an iO which is then run by the decryptor on openings that it computes.

Trading iO *for* ABE. Above, note that the usage of iO is caused by the usage of SSB, which in turn is used to compress L. However, compression of a list has been achieved by much weaker primitives than iO in the literature of broadcast encryption – in particular, the construction of optimal broadcast encryption by Agrawal and Yamada uses the much weaker primitive of ABE (with special efficiency properties) to achieve this. However, ABE does not permit hiding anything other than a message, in particular, an ABE ciphertext cannot encrypt our function f since we desire f to participate in computation. ABE only permits computation on public values, and using ABE to encode f would force f to be public which we cannot allow.

In order to get around this difficulty, we leverage the power of functional encryption (FE), which permits encrypted computation and exactly fills the gap over ABE that we require. A natural candidate for RPE would be to simply use FE to encrypt f, L and m, and encode x and lb in the secret key for a functionality which tests that $\mathsf{lb} \notin L$, that $f(x) = 1$ and outputs m if so. Indeed, this approach using FE is folklore, and was explicitly discussed by NWZ. Yet, they end up with a construction that additionally uses SSB, iO, a puncturable PRF and secret key encryption scheme because of the requirement of size independence from $|L|$ – we do not have candidates for FE with ciphertext size independent of the public attributes. In short, ABE gives us L compactness (in some cases by encoding L in the secret key [BGG+14] and in some cases by encoding L in the ciphertext [ALP11]) but does not hide f, whereas FE gives the opposite.

[4]For the informed reader, this function encodes the "index" and function hiding corresponds to "index hiding" in the literature.

Algebraic Synthesis of ABE *and* FE. We address this conundrum by combining the two primitives in a way that lets us get the best of both. In particular, we use ABE to check that $\mathsf{lb} \notin L$ and use FE to compute $f(x)$. Evidently, the two steps cannot be performed independently in order to resist mix and match attacks so we use nesting, i.e. we use FE to generate ABE ciphertexts. Here, care is required, because ABE encryption takes L as input and done naively, this strategy will again induce a size dependence on L. We address this challenge by using the special ABE by Boneh et al. [BGG+14] which enjoys succinct secret keys and encoding L in the ABE secret key. In more detail, we let the RPE encryption generate ABE.sk(C_L) for a circuit C_L which takes as input lb and checks that $\mathsf{lb} \notin L$. Additionally, it generates an FE ciphertext for the function f and message m. The RPE key generator computes an FE key for a function which has (lb, x) hardwired and takes as input a function f, checks whether $f(x) = 1$ and if so, generates a fresh ABE ciphertext with attribute lb and message m. Thus the decryptor can first compute FE decryption to recover the ABE ciphertext ABE.ct(lb, m) and then use ABE decryption with ABE.sk(C_L) to output m if and only if $\mathsf{lb} \notin L$. It is easy to verify that this construction achieves optimal parameters – this is because ABE has optimal parameters and we used FE only for a simple functionality that does not involve L.

Putting it all Together. The above description is over-simplified and ignores technical challenges such as how to leverage indistinguishability based security of FE, how to generate the randomness used for ABE encryption and such others – we refer the reader to Section 4 for details. However, even having filled in these details, we get only a selectively secure RPE. Substantial work and several new ideas are required for adaptive security, as we discuss next.

Adaptive Security. Next, we outline our ideas to achieve adaptive security, namely where the revocation list L is chosen adaptively by the adversary. Note that to avoid complexity leveraging, we are required to rely only on the selective security of the underling ABE – this creates multiple technical difficulties which are resolved by very carefully using specific algebraic properties of our ingredients.

Leveraging Late Generation of ABE. Our first observation is that full adaptive security of ABE may be unnecessary, since in our construction of RPE, the generation of the ABE instance is deferred until the generation of the challenge ciphertext, at which time the set of revoked users is known. This intuition turns out to be true, but via a complicated security proof as we outline next. Below, we consider the case of function hiding in the RPE ciphertext, the case of message hiding is similar.

Recall that function hiding says that two ciphertexts encoding (f_b, m, L), where $b \in \{0, 1\}$ should be indistinguishable so long as for any requested key $\mathsf{sk}_{\mathsf{lb},x}$ it holds that $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$. Note that the adversary is permitted to query for keys that allow decryption of the ciphertexts, i.e. $f_0(x) = f_1(x) = 1$.

Embedding ABE *CTs in* FE *keys.* In order to use ABE security to prove RPE security, a first (by now standard) step is to use the "trapdoor technique" [CIJ+13, ABSV15, BS18], which allows us to hardwire ABE ciphertexts into FE secret keys. In the security game with the ABE challenger, the reduction submits the label lb associated with each RPE secret key as its challenge attribute and embeds the returned ABE ciphertext into the FE key. Here we immediately run into a difficulty, since in the RPE setting some ABE ciphertexts are decryptable by the adversary and we cannot leverage ABE security. Moreover, we cannot even hope to guess which keys will correspond to decryptable ABE ciphertexts since there are an unbounded polynomial number of key queries in the RPE security game. The same difficulty is faced by NWZ and is the main reason why their construction does not achieve adaptive security in the revocation list.

Polynomial Function Space Suffices for TR. To overcome this hurdle, we leverage the serendipitous fact that for the purpose of constructing TR, it suffices to construct RPE whose function space (recall that functions are encoded in the ciphertext) is only of polynomial size. This observation, which was implicitly present in [GKW19], is abstracted and used explicitly in our proof. In particular, we can assume that the reduction algorithm knows the challenge functions (f_0, f_1) at the beginning of the game, since it can simply guess them. Now, given the secret key query (lb, x), the reduction checks whether $f_0(x) = f_1(x)$. If yes, then there is no need to use ABE security, for the ABE ciphertexts in this case will encode the same message, and will hence be independent of the challenge bit. On the other hand, if $f_0(x) \neq f_1(x)$, then we have by the admissibility condition that $\mathsf{lb} \in L$, even when L is not known. In this case, the reduction can use the security of the ABE without any difficulty.

Additional Hurdles Stemming from ABE *Selective Security.* We now highlight another challenge in the proof. For concreteness, let us consider the second key query $(\mathsf{lb}^{(2)}, x^{(2)})$, which we assume is a pre-challenge query, and assume that $f_0(x^{(2)}) \neq f_1(x^{(2)})$. Hence, by the above discussion, we are required to use ABE security for the ciphertexts with attribute $\mathsf{lb}^{(2)}$. However, according to the selective definition, the reduction is required to choose the challenge attribute at the very start of the game, without even seeing the public parameters. At the same time, the reduction is required to simulate the ABE ciphertext for the first key query, before receiving the second key query from the adversary, that is, without seeing the ABE parameters, leading to an apparent impasse.

We address this issue by considering the following two cases separately: for the first query $(\mathsf{lb}^{(1)}, x^{(1)})$, we have (1) $f_0(x^{(1)}) \neq f_1(x^{(1)})$ or (2) $f_0(x^{(1)}) = f_1(x^{(1)})$. In first case, it is tempting to think that one can simply use a hybrid argument to change the ABE ciphertext associated with each key query satisfying $f_0(x^{(i)}) \neq f_1(x^{(i)})$ for $i \in [2]$. However, this does not work as is, since the ABE ciphertext may leak information about the ABE public key. To address this, we rely on the pseudorandomness of ciphertexts in our ABE [BGG+14] due to which we are guaranteed that the ciphertext does not reveal any information about the public parameters, enabling the hybrid strategy above. To handle the second case, we change the way in which the ABE ciphertext for the first key is generated. In more detail, we stop hardwiring the the ABE ciphertext into the first key and instead generate it directly using ABE parameters. This removes the aforementioned problem since we no longer need to embed the ABE ciphertext or public key into the first FE key. To enable this idea, we introduce additional branch of trapdoor mode for the construction to separate the paths of computation for the cases $f_0(x) = f_1(x)$ and $f_0(x) \neq f_1(x)$. To handle post-challenge queries, we need to address additional challenges, which we do not describe here. We refer the reader to Section 4 for details.

Handling Super-polynomial Revocation List. Our construction (also the secret version, described next) organically supports super-polynomially large revocation list, something that was not known before, to the best of our knowledge. In more detail, let L be a list of super-polynomial size, such that L can be represented as a string of polynomial length and there exists a circuit C_L of polynomial size which takes as input some string lb and checks whether $\mathsf{lb} \in L$ or not. Note that any super-polynomially large list must have efficient representation in order to even allow various algorithms to read it. Then, the key generation of [BGG+14] can naturally encode the circuit C_L as before and the construction works as before. A subtlety that arises with super-polynomial L is that when we deal with post challenge key queries in the proof, we have to deal with the ABE queries in the order of key first and ciphertext later. With polynomial size L, this does not pose a problem because when the adversary chooses L, all the labels for which we use ABE security are in L and we can perform a hybrid argument over these labels. However, this is not possible for super polynomial L, which requires to rely on subexponentially secure

LWE. Please see Section 11 for details.

Instantiating Public RPE. Overall, armed with the above ideas, we get a public RPE from compact FE and efficient ABE supporting exponential sized identity space and adaptive security in the revocation list L. Currently, we only know how to instantiate our desired ABE from LWE [BGG+14], whereas FE can be instantiated in multiple different ways. A natural candidate would be the FE from standard assumptions [JLS21, JLS22] which relies on pairings, LPN and low depth PRG – in this case, our RPE will require the extra assumption of LWE. Another option is to instantiate FE with a post-quantum candidate [GP21, LPST16, WW21, Agr19, DQV+21] from non-standard strengthenings of LWE – this has the advantage that the ABE does not incur any extra assumption in the final construction. For super-polynomial L, we need subexponential hardness of LWE in either pathway to instantiation, as discussed above.

Revocable Predicate Encryption in Private Setting. For private revocable predicate encryption, our starting point is the work of Goyal et al. [GQWW19], who show how to combine "broadcast mixed FE" (called BMFE) together with ABE to achieve RPE (via a different abstraction which they call AugBE). They construct BMFE by adding the broadcast functionality to the primitive of mixed FE defined by [GKW18]. They embed BMFE ciphertext into an ABE ciphertext to achieve RPE, where BMFE is constructed from LWE and ABE is instantiated using pairings.

Supporting Exponential Identity Space. To begin, we upgrade their notion of BMFE to support an exponential space of identities (which we refer to as labels) towards the goal of embedded identity trace and revoke. We refer to our notion as Revocable Mixed FE (denoted by RMFE) and construct it from LWE. Both [GQWW19] and our work start with a mixed FE scheme and add broadcast to it, but their construction builds upon the scheme based on constrained PRFs [CVW+18] while ours begins with the scheme based on Lockable Obfuscation (LO), also from [CVW+18]. Our construction of RMFE deviates significantly from theirs, and achieves significantly better secret key size – $O(\log N)$ as against $O(N)$ – in addition to supporting exponential instead of polynomial space. We describe this construction next.

Mixed FE. The notion of mixed FE was introduced by Goyal, Koppula and Waters in the context of traitor tracing [GKW18]. Identifying and constructing this clever primitive is the key insight that enables [GKW18] to construct traitor tracing with optimal parameters from LWE. Mixed FE is, as the name suggests, a mix of public and secret key FE. Thus, it has a secret as well as a public encryption procedure. The secret encryption procedure takes as input a function f and computes ct_f. This is decryptable by a key sk_x to recover $f(x)$. The adversary can make one query to the encryption oracle in addition to getting the challenge ciphertext for challenge (f_0, f_1). It can also make an unbounded number of key requests so long as $f_0(x) = f_1(x)$. The public encryption algorithm computes a ciphertext for the "always accept" function, i.e. a function which evaluates to 1 for any input x. It is required that the public ciphertext be indistinguishable from the secret ciphertext.

One of the constructions of mixed FE suggested by [CVW+18] uses a secret key FE scheme (SKFE) to construct the secret encryption algorithm and leverages the power of lockable obfuscation (LO) to construct the public encryption procedure. Recall that in a lockable obfuscation scheme [GKW17, WZ17] there exists an obfuscation algorithm Obf that takes as input a program C, a message m and a (random) "lock value" α and outputs an obfuscated program $\tilde{P}$. One can evaluate the obfuscated program on any input x to obtain as output m if $P(x) = \alpha$ and $\perp$ otherwise. Intuitively, the idea of [CVW+18] is to wrap the FE ciphertext using LO and to define the public key encryption algorithm as outputting a simulated version of the LO obfuscated circuit, which is publicly sampleable.

In more detail, the construction works as follows. The secret key for a user with input x is an SKFE secret key $\mathsf{SKFE.sk}(x)$. The secret ciphertext of MFE for function f is constructed as follows.

1. First, SKFE ciphertext $\mathsf{SKFE.ct}(H_{f,\alpha})$ is generated, where α is a freshly chosen random value and $H_{f,\alpha}$ is a circuit that takes as input x and outputs α if $f(x) = 0$ and 0 otherwise.

2. Then, LO with lock value α and any message $m \neq \bot$ is used to obfuscate the circuit $\mathsf{SKFE.Dec}(\mathsf{SKFE.ct}(H_{f,\alpha}), \cdot)$, namely the circuit that takes as input an SKFE secret key and decrypts the hardwired ciphertext using this.

The decryption result of MFE is defined as 1 if the evaluation result of the LO circuit on the given input SKFE secret key is $\bot$ and 0 otherwise. Correctness follows from correctness of SKFE and LO. In particular, if $f(x) = 0$, then SKFE decryption outputs α, which unlocks the LO to give m, otherwise $\bot$. By definition, MFE decryption will output 1 if LO outputs $\bot$ which happens when $f(x) = 1$, and 0 otherwise.

Revocable Mixed FE. RMFE augments MFE so that the encryption algorithms (both secret and public) now include a revocation list L and the secret key additionally includes a label lb. A secret key $\mathsf{sk}_{\mathsf{lb},x}$ decrypts a secret ciphertext $\mathsf{ct}_{f,L}$ to recover $f(x)$ if $\mathsf{lb} \notin L$ and 1 otherwise. For a public ciphertext ct_L, the output of decryption is always 1 regardless of which secret key is being used. For security, we need two properties: function hiding and mode hiding. For function hiding, we require that a secret ciphertext $\mathsf{ct}_{f_0,L}$ is indistinguishable from $\mathsf{ct}_{f_1,L}$ if for all queries, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$. For mode hiding, we require that a secret ciphertext $\mathsf{ct}_{f,L}$ is indistinguishable from a public ciphertext ct_L. Recall that L is not required to be hidden, but we require that the parameters do not depend on $|L|$.

To extend MFE to RMFE, we retain the idea of letting the secret ciphertext be an LO obfuscated circuit and public ciphertext be the simulated LO. To incorporate the list L, we must ensure that the LO lock value α is recovered only when $f(x) = 0$ and $\mathsf{lb} \notin L$. To do so, we consider two subsystems such that one system outputs partial decryption result α_1 only when $f(x) = 0$ and the second system outputs partial decryption result α_2 only when $\mathsf{lb} \notin L$ such that $\alpha = \alpha_1 + \alpha_2$. We must ensure that α_1 and α_2 are user specific decryption results to avoid collusion attacks.

Note that the second subsystem, which entails L, should be constructed so that the hardwired values inside the circuit do not depend on $|L|$, but still control access to the value α_2 depending on L. To satisfy these apparently conflicting requirements, we make use of the unique algebraic properties of the ABE construction by Boneh et al [BGG+14], as described below. For the first subsystem, we use SKFE.

In more detail, our candidate scheme is as follows.

1. **Secret Key:** The RMFE secret key consists of $\mathsf{ABE.ct}(\mathsf{lb}, K)$ and $\mathsf{SKFE.ct}((x, K, R))$ where K and R are user specific random strings, lb is used as an attribute and K is the plaintext for ABE encryption.

2. **Ciphertext:** To generate RMFE ciphertext, the secret key encryption procedure is as follows:

   - It first generates $\mathsf{ABE.sk}(C_L)$, where C_L is a circuit that takes as input a label lb and outputs 1 only when $\mathsf{lb} \notin L$.
   - It also generates $\mathsf{SKFE.sk}(H_{f,\alpha})$, where $H_{f,\alpha}$ takes as input (x, K, R) and outputs $K \oplus \alpha$ if $f(x) = 0$ and R if $f(x) = 1$.

- Now, consider the circuit $CC[\mathsf{ABE.sk}(C_L), \mathsf{SKFE.sk}(H_{f,\alpha})]$, which takes as an input the pair (ABE.ct, SKFE.ct), decrypts both ABE and SKFE ciphertexts using their respective keys, and then outputs the XOR between the decryption results.
- The final ciphertext is an LO of $CC[\mathsf{ABE.sk}(C_L), \mathsf{SKFE.Enc}(H_{f,K,\alpha})]$ with lock value α and any arbitrary message $m \neq \perp$.

By key compactness of [BGG+14], the size of $\mathsf{ABE.sk}(C_L)$ is independent of $|L|$. A subtle point here is that ABE decryption is happening inside the LO and this depends on L. If the LO must process L, then the size of the LO and hence ciphertext blows up with L! Fortunately, the algebraic structure of the ABE scheme we use [BGG+14] again comes to our rescue. At a high level, ABE decryption can be divided into an "L-dependent" step which results in a short processed ciphertext, followed by an "L-independent" step. Importantly, the L-dependent step does not depend on the ABE secret key which is hardwired in the LO and hence inaccessible, and can hence be performed *outside* the LO by the decryptor! The resultant short processed ciphertext can then be provided as input to the LO preventing the problematic size blowup.

RMFE *Proof Overview.* Next we outline some of the ideas developed for the security proof. For ease of understanding, we limit ourselves to the simpler setting where the adversary does not have access to the encryption oracle. This restriction can be removed using combinatorial tricks, similar to [CVW+18]. For security, we must argue two properties – mode indistinguishability and function hiding. The former can be established by relying on security of SKFE and LO analogously to the MFE proof in [CVW+18]. Hence, we focus on function hiding for the rest of the overview, which is subtle and requires several new ideas.

For function hiding, we must make use of the security of ABE and SKFE. Intuitively, security of SKFE guarantees that the values encoded in SKFE ciphertexts and secret keys are hidden, beyond what is revealed by decryption.[5] First note that given a key for (lb, x) such that $f_0(x) = f_1(x)$, no information about the challenge bit is revealed by decryption, since the decryption results of SKFE are the same for both cases. The case with $f_0(x) \neq f_1(x)$ is more challenging. Let us assume $f_0(x) = 0$ and $f_1(x) = 1$. In this case, the decryption result of the challenge ciphertext is R or $K \oplus \alpha$ depending on the value of the challenge bit. Since both are random strings, it is tempting to conclude that they do not reveal any information of the challenge bit.

However, in reality, information about K is encoded in the ciphertext $\mathsf{ABE.ct}(\mathsf{lb}, K)$ and creates a correlation which must be handled. Indeed, a computationally unbounded attacker can learn the challenge bit by breaking open the ABE ciphertext, recovering K and then correlating it with the decryption result of SKFE. Hence, security of ABE must play a role and fortunately, we show that security of ABE suffices to overcome this difficulty. Recall that our security definition of RMFE requires that if $f_0(x) \neq f_1(x)$, then it should hold that $\mathsf{lb} \in L$. This means that the ciphertext $\mathsf{ABE.ct}(\mathsf{lb}, K)$ is computationally indistinguishable from $\mathsf{ABE.ct}(\mathsf{lb}, 0)$, since the only ABE secret key available to the adversary is $\mathsf{ABE.sk}(C_L)$. Now, in the adversary's view, both $K \oplus \alpha$ and R are random strings that are independent from other parameters. Therefore, the adversary cannot obtain any information of the challenge bit from the decryption result in this case as well. For more details, please see Section 5.

Comparison with the BMFE *by Goyal et al. [GQWW19].* We observe that both our RMFE as well as the BMFE by [GQWW19] rely solely on LWE. However, our secret key is $\mathsf{ABE.ct}(\mathsf{lb}, K)$ and $\mathsf{SKFE.ct}((x, K, R))$, which has optimal size, being clearly independent of N and L. In

[5] We note that we need message *and* function hiding security for the underlying SKFE, while [CVW+18] only needs message hiding security.

contrast their secret key depends linearly on N. We also observe that our RMFE can support an exponentially large space of identities, while their BMFE does not.

Combining RMFE *and* ABE *to get* RPE. Finally, we nest our RMFE inside an outer ABE scheme to obtain RPE. This step is very similar to [GQWW19], but we need to use a different ABE scheme. In particular, in the construction of RPE in [GQWW19], a key policy ABE (kpABE) is used to encrypt the message m with attributes as the RMFE ciphertext along with the list L. The RPE secret key for (x, lb) is a kpABE secret key for a the RMFE decryption circuit RMFE.Dec(RMFE.sk, $\cdot, \cdot$).

An obvious difficulty here is that encoding the attribute $(L, \mathsf{RMFE.ct})$ in the ABE ciphertext can cause the ciphertext size to depend on the size of L. To avoid this blowup, [GQWW19] use a special kpABE which has the property that the ciphertext size is independent of the size of the attribute. They instantiate this kpABE with the scheme [ALP11] which uses pairings[6]. However, we cannot use [ALP11, Tak14] because of the following two reasons:

1. First, the ABE scheme by [ALP11] only supports NC_1. However, our circuit RMFE.Dec(RMFE.sk, $\cdot, \cdot$) does not fit into NC_1[7].

2. Furthermore, even if the above problem could be resolved, using [ALP11] is problematic since their ABE has secret and public keys at least as large as $O(|L|)$. While the scheme of [GQWW19] also suffers from this blow-up, our goal is to obtain short keys, independent of $|L|$.

The first problem cannot be resolved even if we use the ABE schemes for circuits [GVW13, BGG$^+$14], since their ciphertext size also depends on $|L|$. To instantiate our ABE, we use recent construction of compact cpABE from evasive and tensor LWE [Wee22], whose parameter sizes depend only on the input length of the circuit and are independent of its size. Armed with the above ideas, we suggest the following RPE:

1. The encryption algorithm of RPE, given m, f, L computes RMFE ciphertext encoding (f, L) and then computes cpABE.Enc(RMFE.Dec(RMFE.ct, $\cdot, L), m$).

2. The key generation algorithm RPE given (lb, x), computes RMFE secret key for (lb, x) and outputs cpABE.sk(RMFE.sk).

Correctness of RPE follows from correctness of cpABE and RMFE while optimality of parameters follows from the efficiency of the underlying schemes. In particular, observe that all parameters are independent of $|L|$. Also note that evasive and tensor LWE are required only to instantiate cpABE with the desired efficiency. If future work standardizes the assumptions underlying the cpABE, our construction will inherit these assumptions. For more details, we refer the reader to Section 6.

Instantiating Secret RPE. Currently, the only two suitable ABE schemes that we know to instantiate our compiler are the LWE based kpABE by Boneh et al. [BGG$^+$14] and the evasive and tensor LWE based cpABE by Wee [Wee22]. These two ABEs give us a secret RPE scheme supporting exponential identities and with optimal parameters, from evasive and tensor LWE. Note that this construction does *not* achieve adaptive security in the revoke list. Nevertheless, it is the first construction of optimal RPE, even without embedded identities, from any assumption

[6] In fact, one could instead use the kpABE constructed by [Tak14]. This enjoys the same efficiency properties and is based on the standard DLIN assumption as against the q-type assumption of [ALP11].

[7] The informed reader may wonder whether we can solve this issue by using preprocessing as in [GQWW19] but this does not work due to technical reasons.

outside Obfustopia. Note that the usage of a non-standard assumption outside of obfustopia (in particular, only from lattice techniques) is somewhat inherent given that even broadcast encryption *without* tracing requires non-standard assumption if we instantiate it only from lattices. We are hopeful that future improvements in cpABE will yield a construction from completely standard assumptions.

Trace and Revoke with Optimal Ciphertext from LWE and Pairings. Along the way, we observe that the broadcast and trace construction provided by Goyal et al. [GQWW19], without embedded identities, can be easily modified to achieve at least optimal ciphertext size, from the same assumptions. At a high level, they construct a broadcast mixed FE from LWE with optimal ciphertext size and then nest this inside the kpABE by [ALP11], which enjoys ciphertext size independent of the attribute length, and can support computation in NC_1. Since their BMFE decryption does not fit into NC_1, they preprocess the ciphertext so that part of the decryption is performed "outside", namely, they group $\log N$ matrix tuples into c groups of $(\log N)/c$ tuples each. Then they precompute all possible $2^{(\log N)/c} = N^{1/c}$ subset-products within each group. Due to this, BMFE decryption only needs to multiply together c of the preprocessed matrices, which can be done in NC_1 so long as c is constant. Unfortunately, this step increases their ciphertext size to $O(N^{\epsilon})$ for any $\epsilon > 0$ though the BMFE ciphertext size was optimal.

We observe that they are "under-using" the ciphertext size independence of [ALP11] – in particular, while the attribute length has indeed been blown up to $O(N^{\epsilon})$, this does not affect the ciphertext size of [ALP11]. Moreover, while the attribute must also be provided outside in the clear, this part can be compressed, i.e. the preprocessing which expands the attribute to size N^{ϵ} can be performed by the decryptor directly by grouping and multiplying matrices as described above, and there is no need for the encryptor to provide this expanded form. Thus, their scheme tweaked with this simple modification already achieves ciphertext of optimal size, though with large secret key $O(N^c)$ for some large constant c.

Trace and Revoke from Revocable Predicate Encryption. It remains to show how to construct the final goal of trace and revoke with embedded identities. As discussed earlier, we follow [NWZ16, KW20] and use the abstraction of RPE to build trace and revoke. However, to embed identities in our trace and revoke schemes, we deviate from these works and instead build upon ideas developed by [GKW19] (henceforth GKW) in the context of traitor tracing.

Embedded Identity Traitor Tracing (EITT) by GKW. The work of Goyal, Koppula and Waters [GKW19] provided an alternative approach for embedding identities in traitor tracing schemes. A well known approach for constructing Traitor Tracing systems suggested by Boneh, Sahai and Waters [BSW07] is via the intermediate primitive of *Private Linear Broadcast Encryption* (PLBE), which allows to construct a tracing algorithm that performs a linear search over the space of users to recover the traitor. Since the number of users was polynomial, this algorithm could be efficient. However, if we allow arbitrary identities to correspond to user indices then the space over which this search must be performed becomes exponential even if the number of users is polynomial, and the trace algorithm is no longer efficient. The main new idea in NWZ that enables them to handle exponentially large identity spaces is to replace a linear search over indices by a clever generalization of binary search, which efficiently solves an "oracle jump problem" which in turn suffices for tracing.

Goyal, Koppula and Waters (GKW) provided an alternate route to the problem of embedding identities. Instead of using PLBE and generalizing the search procedure, they instead extend the definition of PLBE to support embedded identities, denoted by EI-PLBE, and then used this to get a full fledged EITT scheme. This approach has the notable advantage that even if the space of identities is exponential, it can use the fact that the number of users is only *polynomial* and hence rely on only *selective* security of the underlying primitives. In particular, they demonstrate a "nested" tracing approach, where the tracing algorithm works in two steps: first, it outputs a set of indices that correspond to the users that are traitors, and then it uses each index within this set to recover the corresponding identity. Additionally, GKW provide a sequence of (increasingly stronger) TT primitives with embedded identities, namely, indexed EITT, bounded EITT and finally unbounded EITT where unbounded EITT satisfies the most general notion of embedded identity traitor tracing. They also provide generic transformations between these notions, which allows to focus on the weakest notion for any new instantiation.

*Embedded Identity Trace and Revoke (*EITR*).* We adapt the approach of GKW and show how to use their nested approach to trace embedded identities even in the more challenging setting of trace and revoke. As in their case, this lets us use polynomial hardness assumptions in obtaining EITR, in contrast to NWZ. We also define indexed, bounded and unbounded EITR and provide transformations between them. Our definitions as well as transformations are analogous to GKW albeit care is required to incorporate the revoke list L in each step and adapt the definitions and proofs of security accordingly. We then construct indexed EITR using RPE, and obtain unbounded EITR via our generic conversions.

We note that our framework unifies the approaches of Kim and Wu [KW20] who used the framework of RPE in the context of TR and that of GKW who used the framework of EI-PLBE in the context of TT, to obtain EITR. This unification yields a clean abstraction which can be used for both public and secret key settings. We believe this framework is of independent interest. We refer the reader to Sections 7, 8, 9 and 10 for details.

Organization of the paper We provide notation and preliminaries in Section 2. We define RPE in Section 3 and provide a construction of public-key RPE in Section 4. We give our construction of RMFE in Section 5. Then we construct secret-key RPE using RMFE in Section 6. We define different versions of trace and revoke with embedded identities in Section 7 and construct indexed-EITR in Section 8, bounded-EITR in Section 9 and unbounded-EITR in Section 10. Our goal is unbounded-EITR and other variants are introduced as intermediate goals. Secret and public tracing unbounded-EITR can be obtained by applying the conversions in Section 8, 9, and 10 to the secret-key and public key RPE, respectively. We show how to extend our constructions for super-polynomial sized revocation list in Section 11.

2 Preliminaries

In this section we define the notation and preliminaries used in our work.

Notation. We use bold letters to denote vectors and the notation $[a, b]$ to denote the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. Concatenation is denoted by the symbol $\|$. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathsf{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We use the abbreviation PPT for probabilistic polynomial-time. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathsf{negl}(n)$. For two distributions X_λ and Y_λ, $X_\lambda \approx_c Y_\lambda$ denotes that they are computationally indistinguishable for any PPT algorithm. For a vector $\mathbf{x}$, we let x_i denote its i-th entry. For a set S, we let $|S|$ denote the number of elements in S. For a binary string x, we let $|x|$ denote the length of x.

Definition 2.1 (Pseudorandom Functions). A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$, with key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in [\mathbb{N}]}$, domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in [\mathbb{N}]}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in [\mathbb{N}]}$ is a pseudorandom function if it satisfies the following properties:

- **Efficiency:** For all $k \in \mathcal{K}_\lambda$ and $x \in \mathcal{X}_\lambda$, $F(k, x)$ is efficiently computable.
- **Security:** There exists a negligible function negl such that for all PPT adversary $\mathcal{A}$, for all λ, the advantage of $\mathcal{A}$ in the following security experiment is negl.
  $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PRF}}(\lambda)$
  - The challenger samples a key $k \leftarrow \mathcal{K}_\lambda$ and a bit $b \in \{0, 1\}$.
  - The adversary issues polynomially many queries of the following two types in any order:
    Evaluation Queries: $\mathcal{A}$ outputs $x \in \mathcal{X}_\lambda$. The challenger returns $y = F(k, x)$.
    Challenge Queries: $\mathcal{A}$ outputs $x \in \mathcal{X}_\lambda$. The challenger returns y_b, where $y_0 = F(k, x)$ and $y_1 \leftarrow \mathcal{Y}_\lambda$.
  - In the end, $\mathcal{A}$ outputs its guess bit b'.

  $\mathcal{A}$ wins the experiment if $b' = b$.

Remark 2.2. The above security notion is tailored to our purpose and may look stronger than more standard security notion, where the adversary is not allowed to make evaluation queries. However, we can easily show that more standard security notion implies the above by considering the following hybrid games. The first game is the same as above game with $b = 0$. Then, we consider the game where we change all the answers to both evaluation and challenge queries to be random. The game is indistinguishable from the previous one assuming the standard security notion. Finally, we consider a game where answers to the evaluation queries are changed to be $F(k, x)$, while answers to the challenge queries remain random. Again, using the standard security notion, this game is indistinguishable from the previous game. Furthermore, notice that this game is the same as the above game with $b = 1$.

Definition 2.3 (Symmetric Key Encryption). A symmetric key encryption scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in [\mathbb{N}]}$ and key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in [\mathbb{N}]}$ and ciphertext space $\mathcal{CT}_{\mathsf{SKE}}$ has the following syntax:

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

$\mathsf{Enc}(\mathsf{sk}, m) \rightarrow \mathsf{ct}$. The encryption algorithm takes as input the secret key sk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \rightarrow m'$. The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs a message $m' \in \mathcal{M}_\lambda$.

Correctness: A SKE scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

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

Security: A SKE scheme is said to have pseudorandom ciphertext if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\Pr\left[\ \beta' = \beta : \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{sk},\cdot),\mathsf{Enc}^\beta(\mathsf{sk},\cdot)}. \end{array}\ \right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the $\mathsf{Enc}(\mathsf{sk},\cdot)$ oracle, on input a message m, returns $\mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{Enc}(\mathsf{sk},\cdot)$ oracle, on input a message m, returns ct_β, where $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Remark 2.4. We note that similarly to the case of PRF, the above security notion looks a bit different from more standard security notion where the adversary does not have access to $\mathsf{Enc}(\mathsf{sk},\cdot)$ oracle. However, by a similar reduction ro the case of PRF explained in Remark 2.2, it can be seen that they are equivalent.

2.1 Functional Encryption

Here, we recall the definition of public-key and secret-key functional encryption.

2.1.1 Public-Key Functional Encryption

Consider a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, with input space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and output space $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, i.e, $\mathcal{F}_\lambda = \{f : \mathcal{M}_\lambda \to \mathcal{Y}_\lambda\}$. A public-key functional encryption scheme FE for $\mathcal{F}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and outputs a public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_\lambda$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{m}) \to \mathsf{ct}$. The encryption algorithm takes as input the public key mpk and a message $\mathbf{m} \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$. The decryption algorithm takes as input a functional secret key sk_f and a ciphertext ct and outputs $y \in \mathcal{Y}_\lambda$.

Definition 2.5 (Correctness)**.** A functional encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $\mathbf{m} \in \mathcal{M}_\lambda$, and for every function $f \in \mathcal{F}_\lambda$, we have

$$\Pr\Big[\ f(\mathbf{m}) \leftarrow \mathsf{Dec}\big(\mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, \mathbf{m})\big)\ \Big] \geq 1 - \mathsf{negl}(\lambda)$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the probability is taken over the random coins of all algorithms.

Definition 2.6 (Selective Message Privacy)**.** A functional encryption scheme over a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to have selective message privacy (or simply is selectively secure) if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have

$$\Pr\left[\ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{ct}) = b : \begin{array}{l} (\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ b \leftarrow \{0,1\}; \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{m}_b) \end{array}\ \right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where each key query for a function $f \in \mathcal{F}_\lambda$, queried by $\mathcal{A}$ to the KeyGen oracle must satisfy the condition that $f(\mathbf{m}_0) = f(\mathbf{m}_1)$.

Compactness : We say that a FE scheme is compact if the running time of the encryption algorithm only depends on the security parameter and the input message length. In particular, it is independent of the complexity of the function class supported by the scheme.

Definition 2.7 (Fully Compact FE)**.** A functional encryption scheme, $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for input space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and function class $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $\mathcal{F}_\lambda$ is a finite collection of functions, is said to be fully compact if the running time of the encryption algorithm FE.Enc, on input FE.mpk and a message $\mathbf{m} \in \mathcal{M}_\lambda$, is $\mathsf{poly}(\lambda, |\mathbf{m}|)$.

Jain, Lin and Sahai [JLS21, JLS22] provided the first construction of FE with sublinear compactness, from standard assumptions. In more detail:

Lemma 2.8 ([JLS22, GS16, AJS15])**.** *There exists a public-key functional encryption scheme for polynomial sized circuits having selective security (as per Definition 2.6) and full compactness (as per Definition 2.7) with encryption time* $\mathrm{poly}(\lambda, |\mathbf{m}|)$*, where* $\lambda \in \mathbb{N}$ *is the security parameter,* $\mathbf{m}$ *is the input message, assuming LPN, DLIN and existence of boolean PRGs in* NC^0.

2.1.2 Secret Key Functional Encryption

A secret key functional encryption scheme is the same as the public key functional encryption scheme, except that the setup algorithm only outputs msk and the encryption algorithm takes the master secret key msk as input, instead of the master public key mpk.

Definition 2.9 (Ada-IND Function-Message Privacy ([BS18], adapted))**.** A $\mathsf{SKFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ scheme over an input space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to be Ada-IND function and message private if for any PPT algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

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

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$, the $\mathsf{Expt}_\mathcal{A}(\lambda, \mathsf{b})$ is defined as follows

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}_b(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)}$, where the oracle $\mathsf{KeyGen}_b(\mathsf{msk}, \cdot, \cdot)$ on input (f_0, f_1) outputs $\mathsf{KeyGen}(\mathsf{msk}, f_b)$ and $\mathsf{Enc}_b(\mathsf{msk}, \cdot, \cdot)$ on input $(\mathbf{m}_0, \mathbf{m}_1)$ outputs $\mathsf{Enc}(\mathsf{msk}, \mathbf{m}_b)$.
3. Output b'.

and $\mathcal{A}$ is admissible only if for all the key queries $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ and encryption queries $(\mathbf{m}_0, \mathbf{m}_1) \in \mathcal{M} \times \mathcal{M}$, it must hold that $f_0(\mathbf{m}_0) = f_1(\mathbf{m}_1)$.

Note: We refer to the t-bounded Ada-IND function and message private SKFE scheme where Def. 2.9 holds only if at most t queries are made to the $\mathsf{KeyGen}_b(\mathsf{msk}, \cdot, \cdot)$ oracle.

Remark 2.10. We can construct a SKFE scheme satisfying t-bounded Ada-IND message privacy (without function privacy, i.e., $f_0 = f_1$ should hold in the security game) from one-way functions [AV19]. This implies a SKFE scheme satisfying t-bounded Ada-IND function-message privacy, using the conversion results from [BS18].

We also need a 2-bounded semi-adaptive simulation based function-message private SKFE scheme, defined next.

Definition 2.11 (Simulation Based Function-Message Privacy). A secret-key functional encryption scheme is said to satisfy t-bounded semi-adaptive simulation based function and message privacy, if for all PPT stateful algorithm $\mathcal{A}$, there exists PPT stateful algorithms $\mathsf{Sim}^{\mathsf{SK}}, \mathsf{Sim}^{\mathsf{CT}}$ such that:

$$\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathsf{Sim}^{\mathsf{SK}},\mathsf{Sim}^{\mathsf{CT}}}(1^\lambda)\}_{\lambda\in\mathbb{N}}$$

where the real and ideal experiments of stateful algorithms $\mathcal{A}, \mathsf{Sim}^{\mathsf{SK}}, \mathsf{Sim}^{\mathsf{CT}}$ are as follows:

$\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathsf{Sim}}(1^\lambda)$
$\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$
For $i \in [t]$:	For $i \in [t]$:
$\quad \mathcal{A} \to f_i \in \mathcal{F}$;	$\quad \mathcal{A} \to f_i \in \mathcal{F}$;
$\quad \mathcal{A} \leftarrow \mathsf{sk}_{f_i} = \mathsf{KeyGen}(\mathsf{msk}, f_i)$	$\quad \mathcal{A} \leftarrow \mathsf{sk}_{f_i} = \mathsf{Sim}^{\mathsf{SK}}(\mathsf{msk}, 1^{\lvert f_i \rvert})$
Repeat polynomially many times:	Repeat polynomially many times:
$\quad \mathcal{A} \to \mathbf{m} \in \mathcal{M}$;	$\quad \mathcal{A} \to \mathbf{m} \in \mathcal{M}$;
$\quad \mathcal{A} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{m})$	$\quad \mathcal{A} \leftarrow \mathsf{Sim}^{\mathsf{CT}}(\mathsf{msk}, \{f_i(\mathbf{m})\}_{i\in[t]})$
$\mathcal{A} \to b$; Output b	$\mathcal{A} \to b$; Output b

Lemma 2.12. *There exists a 2-bounded semi-adaptive simulation based function and message private* SKFE *scheme, assuming the existence of one-way functions.*

Proof. Let $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ be a 2-bounded $\mathsf{Ada\text{-}IND}$ function-message private SKFE scheme. We construct a 2-bounded semi-adaptive simulation based function-message private $\mathsf{SKFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ scheme as follows:

1. $\mathsf{Setup}(1^\lambda)$: Sample $\mathsf{msk} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$.
2. $\mathsf{KeyGen}(\mathsf{msk}, f)$: $\mathsf{sk} \leftarrow \Pi.\mathsf{KeyGen}(\mathsf{msk}, E[f, 0])$. Here, $E[f, \mathsf{mode}]$ is defined as $E[f, \mathsf{mode}](\mathbf{m}, y_1, y_2) = y_{\mathsf{mode}}$ for $\mathsf{mode} \in \{0, 1, 2\}$, where $y_0 := f(\mathbf{m})$.
3. $\mathsf{Enc}(\mathsf{msk}, \mathbf{m})$: $\mathsf{ct} \leftarrow \Pi.\mathsf{Enc}(\mathsf{msk}, (\mathbf{m}, \perp, \perp)$.
4. $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: $\mathbf{m}' \leftarrow \Pi.\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$.

Simulators for encryption and keygen are described as follows:

$\mathsf{Sim}^{\mathsf{CT}}(\mathsf{msk}, \{f_1(\mathbf{m}), f_2(\mathbf{m})\})$ outputs $\Pi.\mathsf{Enc}(\mathsf{msk}, (\perp, f_1(\mathbf{m}), f_2(\mathbf{m})))$.

$\mathsf{Sim}^{\mathsf{SK}}(\mathsf{msk}, 1^{|f|})$ outputs $\Pi.\mathsf{KeyGen}(\mathsf{msk}, E[1^{|f|}, b])$ for the b-th key query, for $b \in \{1, 2\}$.

Security:
Firstly, observe that the simulator thus constructed is valid since its output does not depend on the function f or input $\mathbf{m}$. Then proof of security follows directly from $\mathsf{Ada\text{-}IND}$ security of Π, because for any set of queries consisting of two functions f_1 and f_2 and polynomial many inputs $\{\mathbf{m}_i\}$, $f_b(\mathbf{m}_i) = g_b(\mathbf{z}_i)$ for $b \in \{1, 2\}$, where g_b is the circuit defined as $(E[1^{|f_b|}, b])$ and $\mathbf{z}_i = (\perp, f_1(\mathbf{m}_i), f_2(\mathbf{m}_i))$ used by the simulators to generate the simulated keys and ciphertexts, respectively. $\square$

2.2 Attribute Based Encryption

We define both ciphertext policy attribute-based encryption (cpABE) and key policy attribute-based encryption (kpABE) in a unified form below.

Let $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_{\lambda \in \mathbb{N}}$ be a relation where A_λ and B_λ denote "ciphertext attribute" and "key attribute" spaces. An attribute-based encryption (ABE) scheme for R and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is defined by the following PPT algorithms:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the unary representation of the security parameter λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{Enc}(\mathsf{mpk}, X, \mu) \to \mathsf{ct}$. The encryption algorithm takes as input a master public key mpk, a ciphertext attribute $X \in A_\lambda$, and a message $\mu \in \mathcal{M}_\lambda$. It outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, Y) \to \mathsf{sk}_Y$. The key generation algorithm takes as input the master secret key msk and a key attribute $Y \in B_\lambda$. It outputs a private key sk_Y.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}, X) \to \mu$ or $\perp$. The decryption algorithm takes as input the master public key mpk, a private key sk_Y, private key attribute $Y \in B_\lambda$, a ciphertext ct and ciphertext attribute $X \in A_\lambda$. It outputs the message μ or $\perp$ which represents that the ciphertext is not in a valid form.

Definition 2.13 (Correctness)**.** An ABE scheme for relation family R is correct if for all $\lambda \in \mathbb{N}$, $X \in A_\lambda, Y \in B_\lambda$ such that $R(X, Y) = 1$, and for all messages $\mu \in \mathcal{M}_\lambda$,

$$\Pr\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X, \mu) : \\ \mathsf{Dec}\Big(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}, X\Big) \neq \mu \end{array}\right] = \mathsf{negl}(\lambda)$$

where the probability is taken over the coins of Setup, KeyGen, and Enc.

Definition 2.14 (Sel-IND security for ABE)**.** For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_{\lambda \in [\mathbb{N}]}$ and a message space $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, let us define Sel-IND security game as follows.

1. $\mathcal{A}$ outputs the challenge ciphertext attribute $X^\star \in A_\lambda$.

2. **Setup phase:** On input 1^λ, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives mpk to $\mathcal{A}$.

3. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order. $\mathcal{A}$ can make unbounded many key queries, but can make only single challenge query.

   (a) **Key Queries:** $\mathcal{A}$ chooses an input $Y \in B_\lambda$. For each such query, the challenger replies with $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y)$.

   (b) **Challenge Query:** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$ to the challenger. The challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$.

   We require that $R(X^\star, Y) = 0$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks.

4. **Output phase:** $\mathcal{A}$ outputs a guess bit b' as the output of the experiment.

We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of $\mathcal{A}$ in the above game as

$$\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 0] - \Pr[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 1]\right|.$$

The ABE scheme ABE is said to satisfy Sel-IND security (or simply *selective security*) if for any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$.

We can consider the following stronger version of the security where we require the ciphertext to be pseudorandom.

Definition 2.15 (Sel-INDr security for ABE)**.** We define Sel-INDr security game similarly to Sel-IND security game except that the adversary $\mathcal{A}$ chooses single message μ instead of (μ_0, μ_1) at the challenge phase and the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu)$ if $b = 0$ and a random ciphertext $\mathsf{ct} \leftarrow \mathcal{CT}$ from a ciphertext space $\mathcal{CT}$ if $b = 1$. Here, we assume that uniform sampling from the ciphertext space $\mathcal{CT}$ is possible without any parameter other than the security parameter λ. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies Sel-INDr security if the quantity is negligible.

We also consider adaptive version of the security.

Definition 2.16 (Ada-IND security for ABE)**.** We define Ada-IND security game similarly to Sel-IND security game except that the adversary $\mathcal{A}$ can choose the challenge ciphertext attribute $X^\star$ adaptively. We define the advantage $\mathsf{Adv}^{\mathsf{Ada\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies Ada-IND security if the quantity is negligible.

In the following, we recall definitions of various ABEs by specifying the relation.

Ciphertext-policy Attribute Based encryption (cpABE). We define cpABE for circuit class $\{\mathcal{C}_{\ell(\lambda),d(\lambda)}\}_\lambda$ by specifying the relation. Here, $\mathcal{C}_{\ell(\lambda),d(\lambda)}$ is a set of circuits with binary output whose input length is $\ell(\lambda)$ and the depth is at most $d(\lambda)$. Note that we do not pose any restriction on the size of the circuits. We define $A^{\mathsf{cpABE}}_\lambda = \mathcal{C}_{\ell(\lambda),d(\lambda)}$ and $B^{\mathsf{cpABE}}_\lambda = \{0,1\}^\ell$. Furthermore, we define the relation $R^{\mathsf{cpABE}}_\lambda$ as

$$R^{\mathsf{cpABE}}_\lambda(C, \mathbf{x}) = C(\mathbf{x}).$$

Key-policy Attribute Based encryption (kpABE). To define kpABE for circuits, we simply swap key and ciphertext attributes in cpABE for circuits. More formally, to define kpABE for circuits, we define $A^{\mathsf{kpABE}}_\lambda = \{0,1\}^\ell$ and $B^{\mathsf{kpABE}}_\lambda = \mathcal{C}_{\ell(\lambda),d(\lambda)}$. We also define $R^{\mathsf{kpABE}}_\lambda : A^{\mathsf{kpABE}}_\lambda \times B^{\mathsf{kpABE}}_\lambda \to \{0,1\}$ as

$$R^{\mathsf{kpABE}}_\lambda(\mathbf{x}, C) = C(\mathbf{x}).$$

Boneh et al. [BGG+14] provided a construction of kpABE which satisfies key compactness and ciphertext succinctness. The following theorem summarizes the efficiency properties of their construction.

Theorem 2.17 (Properties of [BGG+14])**.** *There exists a key-policy ABE scheme* $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Enc}, \mathsf{kpABE.Dec})$ *for function class* $\mathcal{C}_{\ell,d}$ *which is selectively secure under the LWE assumption and has the following properties. In particular:*

Key Compactness. We have $|\mathsf{ABE.sk}_C| \le \mathrm{poly}(\lambda, d)$ *for any* $C \in \mathcal{C}_{\ell,d}$*, where* $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ *and* $\mathsf{ABE.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}, C)$*. In particular, the length of the secret key is independent of the attribute length* ℓ *and the size of the circuit* C*.*

Parameters Succinctness. *We have* $|\mathsf{ABE.mpk}|, |\mathsf{ABE.msk}| \leq \mathrm{poly}(\lambda, d, \ell)$ *and* $|\mathsf{ABE.ct}| \leq \mathrm{poly}(\lambda, d, \ell) + |\mu|$ *for any* $\mathbf{x} \in \mathcal{X}_\lambda$ *and* $\mu \in \mathcal{M}_\lambda$, *where* $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ *and* $\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \mathbf{x}, \mu)$.

Online-Offline Decryption. *The decryption algorithm* $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}_\mathbf{x}, \mathbf{x})$ *can be divided into two parts, which we call*

- $\mathsf{Dec}^{\mathsf{off}}(\mathsf{mpk}, C, \mathbf{x}) \to \mathsf{off}$, *which performs the heavier computation that involves the circuit* C *and attribute* $\mathbf{x}$ *offline without knowing the ciphertext* $\mathsf{ct}_\mathbf{x}$ *or the secret key* sk_C *to get a "help"* off. *We have that the size of* off *is* $\mathrm{poly}(\lambda, \ell, d)$ *and the depth of the circuit* $\mathsf{Dec}^{\mathsf{off}}(\cdot,\cdot,\cdot)$ *is bounded by* $\mathrm{poly}(\lambda, \ell, \mathsf{depth}(C))$, *where* $\mathsf{depth}(C)$ *is the depth of the circuit* C.
- $\mathsf{Dec}^{\mathsf{on}}(\mathsf{sk}_C, \mathsf{ct}_\mathbf{x}, \mathsf{off})$ *takes the help* off *generated offline along with the secret key* sk_C *and the ciphertext* $\mathsf{ct}_\mathbf{x}$ *and outputs the underlying message* μ. *We note that this part does not take* C *as input and in particular, the size of the circuit* $\mathsf{Dec}^{\mathsf{on}}(\cdot,\cdot,\cdot)$ *is* $\mathrm{poly}(\lambda, \ell, d)$, *which is independent from the size of the circuit* C.

We will provide the detail of the kpABE scheme given by [BGG+14] in Sec. 2.3. There, we will show that the scheme satisfies the online-offline decryption property defined above.

We will also use the cpABE scheme given by [Wee22]. The following theorem summarizes the properties of the scheme.

Theorem 2.18 (Properties of [Wee22])**.** *There exists a ciphertext policy ABE scheme* $\mathsf{cpABE} = (\mathsf{cpABE.Setup}, \mathsf{cpABE.KeyGen}, \mathsf{cpABE.Enc}, \mathsf{cpABE.Dec})$ *for function class* $\mathcal{C}_{\ell,d}$, *which is selectively secure under the evasive and tensor LWE assumption and has the following properties. In particular:*

Ciphertext Compactness. *We have* $|\mathsf{cpABE.ct}| \leq \mathrm{poly}(\lambda, d) + |\mu|$ *for any* $C \in \mathcal{C}_{\ell,d}$ *and* $\mu \in \mathcal{M}_\lambda$, *where* $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ *and* $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, \mu)$. *In particular, the size of the ciphertext is independent from the size of the circuit* C *and its input length.*

Parameters Succinctness. *We have* $|\mathsf{cpABE.mpk}|, |\mathsf{cpABE.msk}|, |\mathsf{cpABE.sk}_\mathbf{x}| \leq \mathrm{poly}(\lambda, \ell, d)$ *for any* $\mathbf{x} \in \{0,1\}^\ell$, *where* $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ *and* $\mathsf{cpABE.sk} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathbf{x})$.

2.3 Key-Policy ABE by Boneh et al. [BGG+14]

We will use the kpABE scheme proposed by Boneh et al. [BGG+14]. We provide the description of the scheme in the following while showing that the decryption algorithm can be divided into two phases as stated in Theorem 2.17. We note that the presentation here is largely based on [AY20].

2.3.1 Lattice Preliminaries

Here, we introduce necessary backgrounds for presenting the scheme.

Trapdoors. Let $\mathsf{SampZ}(\gamma)$ be an output of discrete Gaussian distribution with parameter γ over $\mathbb{Z}$. Let us consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. For all $\mathbf{V} \in \mathbb{Z}_q^{n\times m'}$, we let $\mathbf{A}_\gamma^{-1}(\mathbf{V})$ be an output of $\mathsf{SampZ}(\gamma)^{m\times m'}$ conditioned on $\mathbf{A} \cdot \mathbf{A}_\gamma^{-1}(\mathbf{V}) = \mathbf{V}$. A γ-trapdoor for $\mathbf{A}$ is a trapdoor that enables one to sample from the distribution $\mathbf{A}_\gamma^{-1}(\mathbf{V})$ in time $\mathrm{poly}(n, m, m', \log q)$ for any $\mathbf{V}$. We slightly overload notation and denote a γ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\gamma^{-1}$. We also define the special gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ as the matrix obtained by padding $\mathbf{I}_n \otimes (1, 2, 4, 8, \ldots, 2^{\lceil \log q \rceil})$ with zero-columns. The following properties had been established in a long sequence of works. We refer to [AY20] and references therein for the details.

Lemma 2.19 (Properties of Trapdoors)**.** *Lattice trapdoors exhibit the following properties.*

1. *Given* $\mathbf{A}_\tau^{-1}$*, one can obtain* $\mathbf{A}_{\tau'}^{-1}$ *for any* $\tau' \geq \tau$*.*
2. *Given* $\mathbf{A}_\tau^{-1}$*, one can obtain* $[\mathbf{A}\|\mathbf{B}]_\tau^{-1}$ *and* $[\mathbf{B}\|\mathbf{A}]_\tau^{-1}$ *for any* $\mathbf{B}$*.*
3. *There exists an efficient procedure* $\mathsf{TrapGen}(1^n, 1^m, q)$ *that outputs* $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1})$ *where* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *for some* $m = O(n \log q)$ *and is* 2^{-n}*-close to uniform, where* $\tau_0 = \omega(\sqrt{n \log q \log m})$*.*

Lattice Evaluation. The following is an abstraction of the evaluation procedure in previous LWE based FHE and ABE schemes. We follow the presentation by Tsabary [Tsa19].

Lemma 2.20 (Fully Homomorphic Computation [BGG+14])**.** *There exists a pair of deterministic algorithms* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *with the following properties.*

- $\mathsf{EvalF}(\mathbf{B}, F) \to \mathbf{H}_F$*. Here,* $\mathbf{B} \in \mathbb{Z}_q^{n\times m\ell}$ *and* $F : \{0,1\}^\ell \to \{0,1\}$ *is a circuit.*
- $\mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B}) \to \widehat{\mathbf{H}}_{F,\mathbf{x}}$*. Here,* $\mathbf{x} \in \{0,1\}^\ell$ *and* $F : \{0,1\}^\ell \to \{0,1\}$ *is a circuit with depth* d*. We have* $[\mathbf{B} - \mathbf{x} \otimes \mathbf{G}]\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G} \mod q$*, where we denote* $[x_1\mathbf{G}\| \cdots \|x_k\mathbf{G}]$ *by* $\mathbf{x} \otimes \mathbf{G}$*. Furthermore, we have* $\|\mathbf{H}_F\|_\infty \leq m^{O(d)}$ *and* $\|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \leq m^{O(d)}$*.*
- *The running time of* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *is bounded by* $\mathrm{poly}(n, m, \log q, d)$*.*

Note that the last item implies that the circuits computing EvalF and EvalFX can be implemented with depth $\mathrm{poly}(n, m, \log q, d)$.

2.3.2 Key-Policy ABE by Boneh et al. [BGG+14]

The scheme supports the circuit class $\mathcal{C}_{\ell(\lambda),d(\lambda)}$, which is the set of all circuits with input length $\ell(\lambda)$ and depth at most $d(\lambda)$ with arbitrary $\ell(\lambda) = \mathrm{poly}(\lambda)$ and $d(\lambda) = \mathrm{poly}(\lambda)$. In our case, we will set $d(\lambda) = \omega(\log \lambda)$. In the description below, we focus on the case where the message space is $\{0,1\}$ for simplicity. To encrypt a long message, we run the construction in parallel to encrypt an SKE key $K \in \{0,1\}^\lambda$ and then use it to encrypt the message.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, noise distributions χ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, and $B = B(\lambda)$ as specified later. It then proceeds as follows.

1. Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
2. Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_\ell) \leftarrow (\mathbb{Z}_q^{n\times m})^\ell$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
3. Output the master public key $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and the master secret key $\mathsf{msk} = \mathbf{A}_{\tau_0}^{-1}$.

$\mathsf{KeyGen}(\mathsf{msk}, C)$: The key generation algorithm takes as input the master public key mpk, the master secret key msk, and a circuit $C \in \mathcal{C}_{\ell,d}$ and proceeds as follows.

1. Set $F := \neg C$ to be the same circuit as C except that the output bit is flipped. [8]
2. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.

[8] While we follow the standard definition of ABE and require the decryption to be possible when $C(x) = 1$, it is when $C(\mathbf{x}) = 0$ in [BGG+14]. To fill the gap, we flip the output bit.

3. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.
4. Output the secret key $\mathsf{sk}_C := \mathbf{r}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$: The encryption algorithm takes as input the master public key mpk, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$ and proceeds as follows.

1. Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{e}_{i,b} \leftarrow \widetilde{\chi^m}$ for $i \in [\ell]$ and $b \in \{0,1\}$, where $\widetilde{\chi^m}$ is the distribution obtained by first sampling $\mathbf{x} \leftarrow \chi^m$ and $\mathbf{S} \leftarrow \{-1,1\}^{m\times m}$ and then outputting $\mathbf{S}\mathbf{x}$.
2. Compute

$$\text{For all } i \in [\ell],\ b \in \{0,1\}, \psi_{i,b} := \mathbf{s}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{e}_{i,b} \in \mathbb{Z}_q^m$$
$$\psi_{2\ell+1} := \mathbf{s}\mathbf{A} + \mathbf{e} \in \mathbb{Z}_q^m,\ \psi_{2\ell+2} := \mathbf{s}\mathbf{u}^\top + e_0 + \mu\lceil q/2\rceil \in \mathbb{Z}_q,$$

3. Output the ciphertext $\mathsf{ct}_\mathbf{x} := (\{\psi_{i,x_i}\}_{i\in[\ell]}, \psi_{2\ell+1}, \psi_{2\ell+2})$, where x_i is the i-th bit of $\mathbf{x}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}_\mathbf{x}, \mathbf{x})$: The decryption algorithm takes as input the master public key mpk, a secret key sk_C for a circuit C, and a ciphertext $\mathsf{ct}_\mathbf{x}$ for an attribute $\mathbf{x}$ and proceeds as follows. The decryption algorithm can be divided into offline and online phase $\mathsf{Dec}^{\mathsf{off}}$ and $\mathsf{Dec}^{\mathsf{on}}$, respectively, as defined below.

$\mathsf{Dec}^{\mathsf{off}}(\mathsf{mpk}, C, \mathbf{x})$:

1. Compute $C(\mathbf{x})$ and $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalF}(F, \mathbf{x}, \mathbf{B})$, where $F = \neg C$.
2. Output $\mathsf{off} = (C(\mathbf{x}), \widehat{\mathbf{H}}_{F,\mathbf{x}})$.

$\mathsf{Dec}^{\mathsf{on}}(\mathsf{sk}_C, \mathsf{ct}_\mathbf{x}, \mathsf{off})$:

1. Parse off as $(C(\mathbf{x}), \widehat{\mathbf{H}}_{F,\mathbf{x}})$ and $\mathsf{ct}_\mathbf{x} \to (\{\psi_{i,x_i} \in \mathbb{Z}_q^m\}_{i\in[\ell]}, \psi_{2\ell+1} \in \mathbb{Z}_q^m, \psi_{2\ell+2} \in \mathbb{Z}_q)$, and $\mathsf{sk}_C = \mathbf{r} \in \mathbb{Z}^{2m}$. If any of the component is not in the corresponding domain or $C(\mathbf{x}) = 0$, output $\perp$.
2. Concatenate $\{\psi_{i,x_i}\}_{i\in[\ell]}$ to form $\psi_\mathbf{x} = (\psi_{1,x_1}, \ldots, \psi_{\ell,x_\ell})$.
3. Compute
$$\psi' := \psi_{2\ell+2} - [\psi_{2\ell+1}\|\psi_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}]\mathbf{r}^\top.$$
4. Output 0 if $\psi' \in [-B, B]$ and 1 if $[-B + \lceil q/2\rceil, B + \lceil q/2\rceil]$.

Parameters and Security. We choose the parameters for the scheme as follows for concreteness:

$$d = \log n \log\log n, \qquad n = \tilde{\Theta}(\lambda^c), \qquad m = n^{1.1}\log q, \qquad q = n^{O(\log^3 n)},$$
$$\chi = \mathsf{SampZ}(3\sqrt{n}), \qquad \tau_0 = n\log q\log m, \qquad \tau = n^{O(\log^2 n)} \qquad B = n^{O(\log^2 n)},$$

where c is some constant (e.g., $c = 1$).

It is easy to see that the efficiency requirement stated in Theorem 2.17 directly follows from the above parameter settings and Lemma 2.20.

Theorem 2.21 (Adapted from [BGG+14]). *The above scheme satisfies* Sel-INDr *security (Definition 2.15) if we assume the LWE assumption with $n^{O(\log^3 n)}$ approximation factor.*

2.4 Lockable Obfuscation

We define lockable obfusctaion [GKW17, WZ17] below. Consider a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and output space $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$, i.e, $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$. A lockable obfuscation scheme for $\mathcal{F}$ consists of algorithms Obf and Eval with the following syntax:

$\mathsf{Obf}(1^\lambda, f, \alpha) \to \tilde{f}$. The obfuscation algorithm takes as input the security parameter λ, a function $f \in \mathcal{F}_\lambda$ and a lock value $\alpha \in \mathcal{Y}_\lambda$. It outputs an obfuscated program $\tilde{f}$.

$\mathsf{Eval}(\tilde{f}, x) \to 1 \cup \{\perp\}$. The evaluation algorithm takes as input the obfuscated program $\tilde{f}$ and an input $x \in \mathcal{X}_\lambda$. It outputs 1 or $\perp$.

Definition 2.22 (Correctness). A lockable obfuscation scheme is said to be correct if it satisfies the following properties:

1. For all $\lambda \in \mathbb{N}, f \in \mathcal{F}_\lambda, x \in \mathcal{X}_\lambda$ and $\alpha \in \mathcal{Y}_\lambda$ such that $f(x) = \alpha$, we have

$$\mathsf{Eval}\left(\mathsf{Obf}(1^\lambda, f, \alpha), x\right) = 1$$

2. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, f \in \mathcal{F}_\lambda, x \in \mathcal{X}_\lambda$ and $\alpha \in \mathcal{Y}_\lambda$ such that $f(x) \neq \alpha$, we have

$$\Pr[\mathsf{Eval}\left(\mathsf{Obf}(1^\lambda, f, \alpha), x\right) = \perp] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins used during obfuscation.

Definition 2.23 (Security). A lockable obfuscation scheme is said to be secure if there is a PPT simulator Sim such that for all $f \in \mathcal{F}_\lambda$, we have

$$\mathsf{Obf}(1^\lambda, f, \alpha) \approx_c \mathsf{Sim}(1^\lambda, 1^{|f|})$$

where $\alpha \leftarrow \mathcal{Y}_\lambda$ and the probability is taken over the randomness of the obfuscator and simulator Sim.

Theorem 2.24 ([GKW17, WZ17]). *There exists lockable obfuscation for all circuits with lock space $\{0,1\}^\lambda$ from the LWE assumption.*

3 Revocable Predicate Encryption

In this section we define revocable predicate encryption (RPE), in both public and secret key setting. Since the two notions differ only in the encryption algorithm, we present them here in a unified way.

Definition 3.1. A RPE scheme for an attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in [\mathbb{N}]}$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in [\mathbb{N}]}$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda \in [\mathbb{N}]}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in [\mathbb{N}]}$ has the following probabilistic polynomial time algorithms:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes the security parameter λ as input and it outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x) \to \mathsf{sk}_{\mathsf{lb},x}$[9]. The key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}_\lambda$ and an attribute $x \in \mathcal{X}_\lambda$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},x}$.

$\mathsf{Enc}(\mathsf{ek}, f, m, L) \to \mathsf{ct}$. The encryption algorithm takes as input the encryption key ek, a function f, a message $m \in \mathcal{M}_\lambda$, and a revocation list $L \subseteq \mathcal{L}_\lambda$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, \mathsf{ct}, L) \to m'$. The decryption algorithm takes the secret key $\mathsf{sk}_{\mathsf{lb},x}$, a ciphertext ct, and a revocation list L and it outputs $m' \in \mathcal{M}_\lambda \cup \{\perp\}$.

In *public-key RPE*, we take $\mathsf{ek} = \mathsf{mpk}$ in the Enc algorithm, and in *secret-key RPE*, we take $\mathsf{ek} = \mathsf{msk}$.

Furthermore, there is an additional algorithm in the secret key setting defined as follows:

$\mathsf{Broadcast}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. On input the master public key, a message m, and a revocation list $L \subseteq \mathcal{L}_\lambda$, the broadcast algorithm outputs a ciphertext ct.

Definition 3.2 (Correctness)**.** A revocable predicate encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, label $\mathsf{lb} \in \mathcal{L}_\lambda$, attributes $x \in \mathcal{X}_\lambda$, predicates $f \in \mathcal{F}_\lambda$ such that $f(x) = 1$, all messages $m \in \mathcal{M}_\lambda$ and any set of revoked users $L \subseteq \mathcal{L}_\lambda$ such that $\mathsf{lb} \notin L$, if we set $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{sk}_{\mathsf{lb},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x)$, then the following holds

$$\Pr\left[\, \mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, \mathsf{ct}, L) = m \,\right] \geq 1 - \mathsf{negl}(\lambda),$$

for $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, f, m, L)$ (*Encryption correctness*) and $\mathsf{ct} \leftarrow \mathsf{Broadcast}(\mathsf{mpk}, m, L)$ (*Broadcast correctness*).

Security. We now define the security properties of a RPE scheme.

In the following security definitions, we assume for simplicity that the adversary does not make key queries for same input (lb, x) more than once.

Definition 3.3 (q-query Message Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RPE scheme satisfies q-query message hiding property if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$ and any subset of revoked users $L \subseteq \mathcal{L}_\lambda$, the following holds

$$\Pr\left[\ \beta' = \beta : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (f, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot), \mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{ek}, f, m_\beta, L); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot), \mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the encryption oracle $\mathsf{Enc}(\mathsf{ek}, \cdot, \cdot, \cdot)$, and $\mathcal{A}$ is admissible if and only if for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, either $f(x) = 0$ or $\mathsf{lb} \in L$.

Definition 3.4 (q-query Selective Message Hiding)**.** This is the same as the Def 3.3 except that $\mathcal{A}$ outputs the revocation list L in the beginning of the game, before the Setup algorithm is run.

Definition 3.5 (q-query Function Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RPE scheme satisfies q-query function hiding property if for every PPT adversary $\mathcal{A}$, there exists a negligible

[9]We want to point out that the secret key $\mathsf{sk}_{\mathsf{lb},x}$ does not hide the corresponding label lb and attribute x and we assume these to be included in the secret key.

function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$ and any subset of revoked users $L \subseteq \mathcal{L}_\lambda$, the following holds

$$\Pr\left[\beta' = \beta : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (f_0, f_1, m, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{ek}, f_\beta, m, L); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the encryption oracle $\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)$, and $\mathcal{A}$ is admissible if and only if for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)$ oracle, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

Definition 3.6 (q-query Selective Function Hiding)**.** This is the same as the Def 3.5 except that $\mathcal{A}$ outputs the revocation list L in the beginning of the game, before the Setup algorithm is run.

The following security notion is defined only for secret-key RPE scheme.

Definition 3.7 (q-query Selective Broadcast Security)**.** Let $q(\cdot)$ be any fixed polynomial. A RPE scheme satisfies q-query selective broadcast security if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every PPT adversary $\mathcal{A}$, for every $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$ and any subset of revoked users $L \subseteq \mathcal{L}_\lambda$, the following holds

$$\Pr\left[\beta' = \beta : \begin{array}{l} L \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ f, m \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{msk}, f, m, L); \\ \mathsf{ct}_1 \leftarrow \mathsf{Broadcast}(\mathsf{mpk}, m, L); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the encryption $\mathsf{Enc}(\mathsf{msk},\cdot,\cdot,\cdot)$ oracle and $\mathcal{A}$ is admissible if and only if $f(x) = 1, \forall x \in \mathcal{X}_\lambda$.

Remark 3.8. In the *public-key* RPE scheme, the adversary $\mathcal{A}$ can itself simulate the encryption oracle $\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)$, as $\mathsf{ek} = \mathsf{mpk}$ in this setting. Therefore, in public-key setting, we refer to the security definitions without imposing the q-query bound on the encryption oracle.

Remark 3.9. We note that when the message space is binary, function space $\mathcal{F}_\lambda$ is polynomially small and q is a constant, the weaker security definitions where adversary outputs the challenge function f, the challenge message m and the SK-Enc query functions $\{\bar{f}_i\}_{i\in[q]}$ at the beginning of the game, before the $\mathsf{Setup}(1^\lambda)$ algorithm is run, is equivalent to the definitions where the adversary outputs $f, m, \{\bar{f}_i\}_{i\in[q]}$ adaptively. First, the functions can be guessed with polynomial loss. Furthermore, if we restrict the message space to be binary, we can guess the challenge message as well. To extend the message space, we can encrypt each bit by parallel systems.

4 Public-key RPE from FE and LWE

In this section we provide our construction of a public key RPE scheme $\mathsf{RPE} = (\mathsf{RPE.Setup}, \mathsf{RPE.KeyGen}, \mathsf{RPE.Enc}, \mathsf{RPE.Dec})$ for an attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_\lambda$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ from polynomial hardness assumptions. We assume that $|\mathcal{F}_\lambda|$ and $|\mathcal{M}_\lambda|$ are bounded by some polynomial in λ. The restriction on $|\mathcal{F}_\lambda|$ is sufficient for our purpose and the restriction on $|\mathcal{M}_\lambda|$ can be removed by running the scheme in parallel.
Our construction uses the following building blocks:

1. A $\mathsf{Sel\text{-}INDr}$ secure key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.Enc}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Dec})$ for circuit class $\mathcal{C}_{\ell(\lambda),d(\lambda)}$ with parameter succinctness and key compactness (Theorem 2.17). Here $\ell(\lambda)$ is the input length and is the length of labels in our setting and the depth of the circuit is $d(\lambda) \in \omega(\log \lambda)$ to support unbounded revocation list. The message space of the scheme kpABE is $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ and $\mathcal{CT}_{\mathsf{kpABE}}$ denotes the ciphertext space. We assume that uniform sampling from $\mathcal{CT}_{\mathsf{kpABE}}$ is efficiently possible without any parameter.

2. A (fully) compact, selectively secure, public-key functional encryption scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ that supports polynomial sized circuits. We assume that the message space is sufficiently large so that it can encrypt an ABE master public key, a (description of) function $f \in \mathcal{F}_\lambda$, a PRF key, two secret keys of SKE, and a trit $\mathsf{mode} \in \{0, 1, 2\}$.

3. A PRF $F : \{0,1\}^\lambda \times \mathcal{X} \to \{0,1\}^t$ where t is the length of the randomness used in kpABE encryption (Def. 2.1).

4. A symmetric key encryption schemes $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with pseudorandom ciphertexts (Def. 2.3). We let $\mathcal{CT}_{\mathsf{SKE}}$ denote the ciphertext space of SKE.[10] We assume that uniform sampling from $\mathcal{CT}_{\mathsf{SKE}}$ is efficiently possible without any parameter.

We describe our construction below.

$\mathsf{RPE.Setup}(1^\lambda) \to (\mathsf{RPE.mpk}, \mathsf{RPE.msk})$. The setup algorithm does the following:

- Generate $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- Output $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and $\mathsf{RPE.msk} = \mathsf{FE.msk}$.

$\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \mathsf{lb}, x) \to \mathsf{RPE.sk}_{\mathsf{lb},x}$. The key generation algorithm does the following:

- Sample random values $\gamma_1, \gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
- Construct a circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ which has the label lb, attribute x, γ_1, γ_2 and δ hardwired, as defined in Figure 1.
- Compute $\mathsf{FE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta])$.
- Output $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$.

$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m, L) \to \mathsf{RPE.ct}$. The encryption algorithm does the following:

- Parse $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$.
- Sample a PRF key $K \leftarrow \{0,1\}^\lambda$.
- Generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
- Compute $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m, K, 0, \perp, \perp))$.
- Construct a circuit C_L, with revocation list L hardwired defined as follows: On input a label $\mathsf{lb} \in \mathcal{L}_\lambda$,

$$C_L(\mathsf{lb}) = 1 \text{ if and only if } \mathsf{lb} \notin L. \tag{4.1}$$

Compute $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.

[10] We note that we use the same ciphertext space for simplicity even though messages with different lengths are going to be encrypted. To have the same ciphertext space, we can, for example, pad short messages to be some fixed length, which is possible when the message length is bounded by some polynomial.

Function $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$

Hardwired values: A label lb, an attribute x, and SKE ciphertexts γ_1, γ_2, and δ.
Inputs: A kpABE master public key $\mathsf{kpABE.mpk}$, a function $f \in \mathcal{F}_\lambda$, a message $m \in \mathcal{M}_\lambda$, a PRF key K, a trapdoor mode $\mathsf{mode} \in \{0, 1, 2\}$ and SKE keys $\mathsf{SKE.key}_1$ and $\mathsf{SKE.key}_2$.
Output : A kpABE ciphertext.

1. Parse the input as $(\mathsf{ABE.mpk}, f, m, K, \mathsf{mode}, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$.
2. Set $\tilde{m} = \begin{cases} m & \text{if } f(x) = 1 \\ 0 & \text{if } f(x) = 0. \end{cases}$
3. Compute $\mathsf{kpABE.ct}_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.
4. Compute $\mathsf{flag} = \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \delta)$.
5. Compute $\mathsf{out}_i = \mathsf{SKE.Dec}(\mathsf{SKE.key}_i, \gamma_i)$ for $i \in \{1, 2\}$.
6. If $\mathsf{mode} = 0$, output $\mathsf{kpABE.ct}_{\mathsf{lb}}$.
7. If $\mathsf{mode} = 1$, output out_1.
8. If $\mathsf{mode} = 2$, output $\begin{cases} \mathsf{out}_2 & \text{if } \mathsf{flag} = 1 \\ \mathsf{kpABE.ct}_{\mathsf{lb}} & \text{if } \mathsf{flag} = 0. \end{cases}$

Figure 1: Function to compute kpABE ciphertexts depending on various conditions.

- Output $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$.

$\mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},x}, \mathsf{RPE.ct}, L) \to m'$. The decryption algorithm does the following:

- Parse $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ and $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$.
- Compute $\mathsf{ct}' = \mathsf{FE.Dec}(\mathsf{FE.sk}_{\mathsf{lb},x}, \mathsf{FE.ct})$.
- Construct circuit C_L from L and compute $m' = \mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, C_L, \mathsf{ct}', \mathsf{lb})$.
- Output m'.

Correctness. We now show that the above construction is correct via the following theorem.

Theorem 4.1. *Suppose* FE *and* kpABE *schemes are correct. Then the above construction satisfies the encryption correctness (Def. 3.2).*

Proof. Firstly, for any label lb, attribute x, and function f such that $f(x) = 1$, we have $\mathsf{FE.Dec}(\mathsf{FE.sk}_{\mathsf{lb},x}, \mathsf{FE.ct}) = \mathsf{kpABE.ct}_{\mathsf{lb}}$, where $\mathsf{kpABE.ct}_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, m; F(K, (\mathsf{lb}, x)))$, by the correctness of FE and the definition of Re-Enc. We then observe that $C_L(\cdot)$ can be implemented with depth $O(\log(|L| \cdot |\mathsf{lb}|)) = O(\log \mathrm{poly}(\lambda)) = O(\log \lambda) \le d$ and thus $C_L(\cdot) \in \mathcal{C}_{\ell,d}$. Then if $\mathsf{lb} \notin L$ we have $C_L(\mathsf{lb}) = 1$ and hence from the correctness of the kpABE scheme it follows that $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, C_L, \mathsf{kpABE.ct}_{\mathsf{lb}}, \mathsf{lb}) = m$. So the decryption correctly recovers the message when $f(x) = 1$ and $\mathsf{lb} \notin L$. □

Efficiency. Here we argue that our construction achieves optimal parameters. Namely, we show that the size of each parameter is independent from $|L|$. We note that $|f|$ refers to the description size of the function, not the circuit size that implements the function. When $|x|$ is very long and f has succinct description, the former can be much shorter than the latter.

1. Public key size $|\mathsf{RPE.mpk}|$: We have $|\mathsf{RPE.mpk}| = |\mathsf{FE.mpk}|$. Since we assumed that FE is fully compact (Def. 2.7), the length of FE.mpk only depends on the input length of Re-Enc. We have that the input length is $|\mathsf{ABE.mpk}| + |f| + |m| + |K| + |\mathsf{mode}| + 2|\mathsf{SKE.key}| = |\mathsf{ABE.mpk}| + |f| + O(\lambda)$. We have $|\mathsf{ABE.mpk}| \leq \mathrm{poly}(\lambda, |\mathsf{lb}|, d) = \mathrm{poly}(\lambda, |\mathsf{lb}|)$ by Theorem 2.17. The total length is therefore $\mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$.

2. Secret key size $|\mathsf{RPE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RPE.sk}_{\mathsf{lb},x}| = |\mathsf{FE.sk}_{\mathsf{lb},x}|$. Since the size of the latter is polynomially dependent on the size of Re-Enc, we evaluate its size. We can see that the size of Re-Enc is polynomial in the total length of the input and the hardwired values. The length of the input is bounded by $\mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$ as analyzed in the above item. The length of the hardwired values are $|\mathsf{lb}|+|x|+|\gamma_1|+|\gamma_2|+|\delta|$. We have $|\gamma_1|+|\gamma_2|+|\delta| = 3|\gamma_2|$[11] and $|\gamma_2| = \mathrm{poly}(\lambda, \mathsf{kpABE.ct}_{\mathsf{lb}}) = \mathrm{poly}(\lambda, \mathsf{lb}, d) = \mathrm{poly}(\lambda, \mathsf{lb})$. Therefore, the size of Re-Enc is $\mathrm{poly}(\lambda, |x|, |f|, |\mathsf{lb}|)$ and so is the size of the secret key.

3. Ciphertext size $|\mathsf{RPE.ct}|$: We have $|\mathsf{RPE.ct}| = |\mathsf{kpABE.mpk}| + |\mathsf{kpABE.sk}_L| + |\mathsf{FE.ct}|$. We have $|\mathsf{ABE.mpk}| = \mathrm{poly}(\lambda, |\mathsf{lb}|)$ as we showed in the first item. We also have $|\mathsf{kpABE.sk}_L| \leq \mathrm{poly}(\lambda, d) \leq \mathrm{poly}(\lambda)$ by the key compactness of kpABE (2.17). By similar analysis to the first item, full compactness of FE implies $|\mathsf{FE.ct}| \leq \mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$. Therefore, the overall length of the ciphertext is $\mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$.

Security. Now we prove that the above construction of RPE satisfies both function hiding and message hiding security.

Function Hiding

Theorem 4.2. *Assume that F is a secure* PRF, SKE *is correct and secure,* FE *and* kpABE *are secure as per definitions 2.6 and 2.15, respectively. Furthermore, assume $|\mathcal{F}_\lambda| \leq \mathrm{poly}(\lambda)$ and $|\mathcal{M}_\lambda| \leq \mathrm{poly}(\lambda)$. Then the* RPE *constructed above is function hiding (Def. 3.5).*

Proof. Recall that for function hiding we need $\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_0, m, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_1, m, L)$, where for all the key queries (lb, x), either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

The proof proceeds via a sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$.

Hybrid_0. This is the real world with $\beta = 0$, i.e. the challenge ciphertext is computed using the function f_0. We write the complete game here to set up the notations and easy reference in later hybrids.

1. The adversary outputs the challenge functions f_0 and f_1 and the challenge message m[12].

[11] We assumed that $|\gamma_1| = |\gamma_2| = |\delta|$. See Footnote 10.

[12] To keep the proofs and notations simple, we let f_0, f_1, m to be given selectively. This is sufficient to achieve security as in Def 3.5, as mentioned in Remark 3.9.

2. The challenger generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends it to the adversary. The challenger then responds to different queries from $\mathcal{A}$ as follows:
3. **Key Queries :** For each key query (lb, x), the challenger does the following:
   - Samples random values $\gamma_1, \gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Defines the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ as in Figure 1 and computes $\mathsf{FE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta])$.
   - Returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$ to the adversary.
4. **Challenge Query:** When the adversary outputs the revocation list L for the challenge query, the challenger does the following:
   - Samples a PRF key K.
   - Generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
   - Computes FE.ct as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp)).$$

   - Defines C_L as in Eq. (4.1) and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to the adversary.
5. In the end, the adversary outputs a bit β'.

Hybrid_1. This hybrid is same as the previous hybrid except the following changes:

- The challenger generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ in the beginning of the game after the adversary outputs the challenge functions f_0 and f_1 and the challenge message m. It also samples a SKE secret key $\mathsf{SKE.key}_1$ and a PRF key K.
- For each key query (lb, x), γ_1 is computed differently. In particular, the challenger does the following
  - Sets $\tilde{m} = \begin{cases} m & \text{if } f_0(x) = 1 \\ 0 & \text{if } f_0(x) = 0 \end{cases}$
    and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.
  - Sets γ_1 as $\mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hybrid_2. This hybrid is same as the previous hybrid except the following changes:

- The challenger samples two SKE secret keys $\mathsf{SKE.key}_1$ and $\mathsf{SKE.key}_2$ in the beginning of the game (after receiving f_0, f_1, m from $\mathcal{A}$).
- For each key query (lb, x), γ_2 and δ are computed differently from the previous hybrid. In particular, the challenger does the following:
  - Sets $\mathsf{flag} = \begin{cases} 1 & \text{if } f_0(x) \neq f_1(x) \\ 0 & \text{otherwise.} \end{cases}$
  - Sets $\delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. We note that $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ and γ_1 are computed as in the previous hybrid.

Hybrid_3. This hybrid is same as the previous hybrid except that FE.ct in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hybrid_4. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as follows:

- If $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$, where $r \leftarrow \{0,1\}^t$.
- Else, if $\mathsf{flag} = 0$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

Hybrid_5. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)).$$

Hybrid_6. This hybrid is same as the previous hybrid except that for each key query (lb, x), γ_1 is set differently as $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hybrid_7. This hybrid is same as the previous hybrid except that for each such key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is sampled uniformly from $\mathcal{CT}_{\mathsf{kpABE}}$, i.e., $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.

Hybrid_8. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_1, m, K, 2, \perp, \mathsf{SKE.key}_2)).$$

We note that in the hybrids hereafter, we rewind the changes made in the preceding hybrids.

Hybrid_9. This hybrid is same as the previous hybrid except that for each such key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is changed back to $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$ for $r \leftarrow \{0,1\}^t$, where $\tilde{m}$ is now defined as m if $f_1(x) = 1$ and 0 otherwise.

Hybrid_{10}. This hybrid is same as the previous hybrid except that for all the key queries (lb, x), the challenger sets γ_1 as $\mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hybrid_{11}. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_1, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hybrid_{12}. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

Hybrid_{13}. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_1, m, K, 0, \perp, \perp)).$$

Hybrid_{14}. This hybrid is same as the previous hybrid except that γ_2 and δ_2 are set as $\gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hybrid_{15}. This hybrid is same as the previous hybrid except that γ_1 is set as $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. Note that this is the real world with $\beta = 1$, i.e., f_1 is encrypted in the challenge ciphertext.

Indistinguishability of hybrids We now show that the consecutive hybrids are indistinguishable.

Claim 4.2.1. *Assume that* SKE *is secure, then* $\mathsf{Hybrid}_0 \approx_c \mathsf{Hybrid}_1$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hybrid_0 and Hybrid_1 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of SKE scheme with advantage ϵ. The reduction is as follows.

1. The SKE challenger samples $\mathsf{SKE.key}_1 \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and a bit $\hat{\beta} \leftarrow \{0,1\}$ and starts the SKE security game with $\mathcal{B}$.

2. $\mathcal{B}$ invokes $\mathcal{A}$, which then outputs the challenge functions f_0 and f_1 and the challenge message m.

3. $\mathcal{B}$ generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends $\mathsf{RPE.mpk}$ to $\mathcal{A}$.
   $\mathcal{B}$ also generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and samples a PRF key K.

4. **Key Queries :** Whenever $\mathcal{A}$ issues a key query (lb, x), $\mathcal{B}$ does the following:

   - It sets $\tilde{m} = \begin{cases} m & \text{if } f_0(x) = 1 \\ 0 & \text{if } f_0(x) = 0 \end{cases}$
     and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.
   - It sends $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ as the challenge message to the SKE challenger. The SKE challenger returns $\mathsf{ct}_{\hat{\beta}}$ to $\mathcal{B}$, where $\mathsf{ct}_0 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ and $\mathsf{ct}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.
   - It sets $\gamma_1 = \mathsf{ct}_{\hat{\beta}}$, samples random values $\gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ and computes the FE key for the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

5. **Challenge Query :** When $\mathcal{A}$ outputs the revocation list L for the challenge ciphertext, $\mathcal{B}$ does the following:

   - Computes $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp))$.
   - Defines C_L as in Eq. 4.1 and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ sends β' to the SKE challenger.

We observe that if the SKE challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hybrid_0, else Hybrid_1 with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_0) - \Pr(\beta' = 1|\mathsf{Hybrid}_1)|$ = ϵ (by assumption).

□

Claim 4.2.2. *Assume that* SKE *is secure. Then* $\mathsf{Hybrid}_1 \approx_c \mathsf{Hybrid}_2$.

Proof. The proof follows the same steps as that for the claim 4.2.1 and hence omitted. □

Claim 4.2.3. *Assume that* FE *satisfies selective security (Def. 2.6) and* SKE *is correct. Then* $\mathsf{Hybrid}_2 \approx_c$ *and* Hybrid_3.

Proof. We show that if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of FE with the same advantage ϵ. $\mathcal{B}$ is defined as follows:

1. Upon being invoked by the FE challenger, $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge functions f_0 and f_1 and the challenge message m.

2. $\mathcal{B}$ samples two SKE secret keys $\mathsf{SKE.key}_1, \mathsf{SKE.key}_2$, a PRF key K and generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.

3. It sets $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$ and sends (μ_0, μ_1) to the FE challenger as challenge messages.

4. The FE challenger generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, samples $\hat{\beta} \leftarrow \{0,1\}$. It then computes $\mathsf{FE.ct}_{\hat{\beta}} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, \mu_{\hat{\beta}})$ and sends $(\mathsf{FE.mpk}, \mathsf{FE.ct}_{\hat{\beta}})$ to $\mathcal{B}$.

5. $\mathcal{B}$ sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends $\mathsf{RPE.mpk}$ to $\mathcal{A}$.

6. **Key Queries :** When $\mathcal{A}$ issues a key query (lb, x), $\mathcal{B}$ does the following:

   - It sets $\tilde{m} = \begin{cases} m & \text{if } f_0(x) = 1 \\ 0 & \text{if } f_0(x) = 0 \end{cases}$
     and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.
   - It sets $\mathsf{flag} = \begin{cases} 1 & \text{if } f_0(x) \neq f_1(x) \\ 0 & \text{otherwise} \end{cases}$.
   - It computes $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}}), \delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else samples $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - It defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ as in Figure 1 and sends a key query $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ to the FE challenger. The FE challenger returns the secret key $\mathsf{FE.sk}_{\mathsf{lb},x}$ to $\mathcal{B}$.
   - $\mathcal{B}$ returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

7. **Challenge Query :** When $\mathcal{A}$ outputs the revocation list L for the challenge ciphertext, $\mathcal{B}$ does the following:

   - It defines C_L as in Eq. (4.1) and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct}_{\hat{\beta}})$ to $\mathcal{A}$.

8. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ sends β' to the FE challenger.

We observe that if FE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hybrid_2, else Hybrid_3 with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_2) - \Pr(\beta' = 1|\mathsf{Hybrid}_3)|$ = ϵ (by assumption).

Admissibility of $\mathcal{B}$

Firstly, we observe that the only key queries that $\mathcal{B}$ issues to the FE challenger are for the Re-Enc functions defined for each key query (lb, x) by $\mathcal{A}$. Next, we observe that for any function $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp)$, and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$ the following holds true from the definition of Re-Enc and correctness of SKE decryption,

$$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) = \mathsf{kpABE.ct}_{\mathsf{lb}}$$

$$= \quad \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$$

$$\begin{aligned}
\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) &= \mathsf{SKE.Dec}(\mathsf{SKE.key}_1, \gamma_1) \\
&= \mathsf{kpABE.ct}'_{\mathsf{lb}} \\
&= \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))
\end{aligned}$$

Thus, for all the keys queried to the FE challenger, $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) = \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1)$. This establishes the admissibility of $\mathcal{B}$. □

Claim 4.2.4. *Assume that* PRF *is secure, then* $\mathsf{Hybrid}_3 \approx_c \mathsf{Hybrid}_4$.

Proof. We show that if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against PRF security with the same advantage ϵ. The reduction $\mathcal{B}$ is defined as follows:

1. The PRF challenger samples a PRF key K and a bit $\hat{\beta} \leftarrow \{0, 1\}$ and starts the game with $\mathcal{B}$.
2. $\mathcal{B}$ then invokes $\mathcal{A}$ which outputs the challenge functions f_0 and f_1 and the challenge message m.
3. $\mathcal{B}$ samples two SKE secret keys $\mathsf{SKE.key}_1, \mathsf{SKE.key}_2$, generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends $\mathsf{RPE.mpk}$ to $\mathcal{A}$.
4. **Key Queries :** When $\mathcal{A}$ issues a key query (lb, x), $\mathcal{B}$ does the following:
   - If $\mathsf{flag} = 0$, it sends an evaluation query for input (lb, x) to the PRF challenger and gets back $F(K, (\mathsf{lb}, x))$. It then computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.
   - If $\mathsf{flag} = 1$, it sends (lb, x) as a challenge query to the PRF challenger and gets back $r_{\hat{\beta}}$, where $r_0 = F(K, (\mathsf{lb}, x))$ and $r_1 \leftarrow \{0, 1\}^t$. It then computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r_{\hat{\beta}})$.
   - Computes $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}}), \delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and computes the FE key for the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.
5. **Challenge Query :** When $\mathcal{A}$ outputs the revocation list L for the challenge ciphertext, $\mathcal{B}$ does the following
   - Computes $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2))$.
   - Defines C_L as in Eq. 4.1 and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.
6. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ sends β' to the SKE challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hybrid_3, else Hybrid_4 with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_3) - \Pr(\beta' = 1|\mathsf{Hybrid}_4)|$ = ϵ (by assumption).

□

Claim 4.2.5. *Assume that* FE *is selectively secure (as per Def 2.6) and* SKE *is correct. Then* $\mathsf{Hybrid}_4 \approx_c \mathsf{Hybrid}_5$.

Proof. We show that if $\mathcal{A}$ wins with non-negligible advantage ϵ in distinguishing the two hybrids, then there exists an adversary $\mathcal{B}$ against FE security with the same advantage ϵ. The steps of the reduction are similar as in the proof of the Claim 4.2.3, with $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)$. Hence, here we only argue the admissibility of $\mathcal{B}$ in the FE security game.

Admissibility of $\mathcal{B}$

Firstly, we observe that the only key queries that $\mathcal{B}$ issues to the FE challenger are for the $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ functions, where each function is defined corresponding to a key query (lb, x) by $\mathcal{A}$. Here, $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}}), \delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. Also $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$ if $\mathsf{flag} = 0$; else $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$ where $r \leftarrow \{0,1\}^t$.

Next, we observe that for any function $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$, and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)$ the following holds true from the definition of Re-Enc and correctness of SKE decryption,

$$\begin{aligned}
\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) &= \mathsf{SKE.Dec}(\mathsf{SKE.key}_1, \gamma_1) \\
&= \mathsf{kpABE.ct}'_{\mathsf{lb}} \\
&= \begin{cases} \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x))) & \text{if } \mathsf{flag} = 0 \\ \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r) & \text{if } \mathsf{flag} = 1 \end{cases}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) &= \begin{cases} \mathsf{kpABE.ct}_{\mathsf{lb}} & \text{if } \mathsf{flag} = 0 \\ \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \gamma_2) = \mathsf{kpABE.ct}'_{\mathsf{lb}} & \text{if } \mathsf{flag} = 1 \end{cases} \\
&= \begin{cases} \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x))) & \text{if } \mathsf{flag} = 0 \\ \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r) & \text{if } \mathsf{flag} = 1 \end{cases}
\end{aligned}$$

Thus, $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0)$=$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1)$. This establishes the admissibility of $\mathcal{B}$. □

Claim 4.2.6. *Assume that* SKE *is secure, then* $\mathsf{Hybrid}_5 \approx_c \mathsf{Hybrid}_6$.

Proof. The proof is similar to the proof for the claim 4.2.1 and hence omitted. □

Claim 4.2.7. *Assume that* kpABE *is* $\mathsf{Sel\text{-}INDr}$ *secure (Def. 2.15), then* $\mathsf{Hybrid}_6 \approx_c \mathsf{Hybrid}_7$.

Proof. To argue indistinguishability between the two hybrids, we define an intermediate hybrid Hybrid_{6a} as follows - this hybrid is same as Hybrid_6 except that for each such pre-challenge key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. The proof then follows from the following two claims:

Claim 4.3 (1). *Assuming* kpABE *is* $\mathsf{Sel\text{-}INDr}$ *secure (Def. 2.15),* $\mathsf{Hybrid}_6 \approx_c \mathsf{Hybrid}_{6a}$.

Proof. Let Q_{fpre} be the number of pre-challenge key queries with $\mathsf{flag} = 1$[13]. Then, we further define the following sub hybrids: for $i = 0$ to Q_{fpre}, define $\mathsf{Hybrid}_{6.i}$ which is same as Hybrid_6,

[13]Note that we can upper bound Q_{fpre} as $Q_{\mathsf{fpre}} \leq |L|$.

except that for the first i key queries with $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Thus, $\mathsf{Hybrid}_{6.0} = \mathsf{Hybrid}_6$ and $\mathsf{Hybrid}_{6.Q_{\mathsf{fpre}}} = \mathsf{Hybrid}_{6a}$. Next, we show that for all $i \in [Q_{\mathsf{fpre}}]$, $\mathsf{Hybrid}_{6.i-1} \approx_c \mathsf{Hybrid}_{6.i}$. In particular, we show that if $\mathcal{A}$ distinguishes between the two hybrids with non-negligible advantage ϵ, then there exists a PPT algorithm $\mathcal{B}$ against Sel-INDr security of kpABE with the same advantage ϵ.

Observe that the two hybrids differ only in the value of $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ used in the computation of $\mathsf{RPE.sk}_{\mathsf{lb},x}$ for the i-th key query (lb, x) with $\mathsf{flag} = 1$; in the former hybrid we have $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$, while in the latter hybrid, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Now, we define the reduction $\mathcal{B}$.

1. $\mathcal{B}$ firstly invokes $\mathcal{A}$ and gets f_0, f_1 and m.
2. $\mathcal{B}$ then samples two SKE secret keys $\mathsf{SKE.key}_1, \mathsf{SKE.key}_2$, a PRF key K and generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends it to $\mathcal{A}$. It then answers different queries from $\mathcal{A}$ as follows:
3. **Key Queries**: For each key query (lb, x), $\mathcal{B}$ does the following:
   - Sets $\mathsf{flag} = 0$, if $f_0(x) = f_1(x)$; else $\mathsf{flag} = 1$. It computes $\delta = \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and samples $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Computes γ_2 as follows:
     - If $\mathsf{flag} = 0$, $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
     - Else, if $\mathsf{flag} = 1$, then let this be the j-th key query with $\mathsf{flag} = 1$. Then,
       * For $j < i$, $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. (Note that this does not require kpABE.mpk).
       * For $j = i$, $\mathcal{B}$ does the following:
         · Sends lb and $\tilde{m}$ as the challenge attribute and message, respectively, to the kpABE challenger.
         · The kpABE challenger samples $\hat{\beta} \leftarrow \{0, 1\}$ and computes $\mathsf{kpABE.ct} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$, if $\hat{\beta} = 0$, else samples $\mathsf{kpABE.ct} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. The kpABE challenger sends $\{\mathsf{kpABE.mpk}, \mathsf{kpABE.ct}\}$ to $\mathcal{B}$.
         · $\mathcal{B}$ then computes $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct})$.
       * For $j > i$, $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$.
   - Defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, computes an FE key for this circuit and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.
4. **Challenge Query:** When $\mathcal{A}$ outputs the revocation list L for the challenge query, $\mathcal{B}$ does the following:
   - Computes $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2))$.
   - Defines C_L as in Eq. 4.1 and sends a key query for the circuit C_L to the kpABE challenger. The kpABE challenger returns $\mathsf{kpABE.sk}_L$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.
5. In the end, $\mathcal{A}$ outputs its guess bit β'. $\mathcal{B}$ sends β' to the kpABE challenger as its guess bit.

We observe that if the kpABE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated $\mathsf{Hybrid}_{6,i-1}$, else $\mathsf{Hybrid}_{6,i}$ with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)| =$

$|\Pr(\beta' = 1|\mathsf{Hybrid}_{6,i-1}) - \Pr(\beta' = 1|\mathsf{Hybrid}_{6,i})| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$: Observe that $\mathcal{B}$ issues a single kpABE key query which is for circuit C_L and the challenge attribute is the label lb corresponding to a key query (lb, x) by $\mathcal{A}$ for which flag $= 1$. Hence, by the admissibility condition of $\mathcal{A}$, $\mathsf{lb} \in L$ and hence $C_L(\mathsf{lb}) = 0$. □

Claim 4.4 (2). *Assume* kpABE *is* Sel-INDr *secure (Def. 2.15). Then,* $\mathsf{Hybrid}_{6a} \approx_c \mathsf{Hybrid}_7$.

Proof. To prove the claim, we again consider sub hybrids, $\mathsf{Hybrid}_{6a.i}$ for $i = 0$ to $|L|$, defined as follows: let $L_{[1:j]}$ be the set of first j labels in the revocation list L. Then, $\mathsf{Hybrid}_{6a.i}$ is same as Hybrid_{6a} except the following changes: for any post-challenge key query (lb, x) such that $\mathsf{lb} \in L_{[1:i]}$ and flag $= 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Thus, $\mathsf{Hybrid}_{6a.0} = \mathsf{Hybrid}_{6a}$ and $\mathsf{Hybrid}_{6a.|L|} = \mathsf{Hybrid}_7$. Next we prove the following claim:

Claim 4.5. *Assume* kpABE *is* Sel-INDr *secure (Def. 2.15). Then for* $i \in [|L|]$, $\mathsf{Hybrid}_{6a.i-1} \approx_c \mathsf{Hybrid}_{6a.i}$.

Proof. Let lb_i be the i-th label in L. Then we observe that if there is no post-challenge key query (lb, x) such that flag $= 1$ and $\mathsf{lb} = \mathsf{lb}_i$ then the two hybrids are identical. Else, we show that if $\mathcal{A}$ can distinguish between the two hybrids with non negligible advantage ϵ then there exists a PPT algorithm $\mathcal{B}$ against Sel-INDr security of kpABE security with the same advantage ϵ. $\mathcal{B}$ is defined as follows:

1. $\mathcal{B}$ firstly invokes $\mathcal{A}$ and gets f_0, f_1 and m.
2. It then samples a SKE secret key $\mathsf{SKE.key}_2$, a PRF key K and generates FE keys as $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends it to $\mathcal{A}$. It then answers different queries from $\mathcal{A}$ as follows:
3. **Pre-challenge Key Queries:** For pre-challenge key query (lb, x), $\mathcal{B}$ does the following:
   - Sets flag $= 0$ if $f_0(x) = f_1(x)$; else flag $= 1$ and computes $\delta = \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$. It also samples $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Computes γ_2 as follows: if flag $= 0$, $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$; else $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. (Note that this does not require kpABE.mpk).
   - Defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, computes an FE key for this circuit and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.
4. **Challenge Query:** When $\mathcal{A}$ outputs the revocation list $L = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_{|L|}\}$ for the challenge ciphertext, $\mathcal{B}$ does the following:
   - Sends lb_i to the kpABE challenger as the challenge attribute. The kpABE challenger samples $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and $\hat{\beta} \leftarrow \{0,1\}$, and sends kpABE.mpk to $\mathcal{B}$.
   - Constructs circuit C_L as defined in the construction and sends a key query for C_L to the kpABE challenger and gets $\mathsf{kpABE.sk}_L$ in response.
   - Computes $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2))$
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.

5. **Post-challenge Key Queries:** For each post-challenge key query (lb, x), $\mathcal{B}$ computes flag, δ and γ_1 as defined for the hybrid and defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, where γ_2 is computed as follows:

   - if $\mathsf{flag} = 0$, $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. Else,
     - if $\mathsf{lb} \in L_{[1:i-1]}$, $\gamma_2 = \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.
     - if $\mathsf{lb} = \mathsf{lb}_i$, $\mathcal{B}$ sends challenge query with message $\tilde{m}$ to the kpABE challenger. The kpABE challenger returns a ciphertext $\mathsf{kpABE.ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}_i, \tilde{m}; r)$, if $\hat{\beta} = 0$; else $\mathsf{kpABE.ct} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Then $\mathcal{B}$ computes $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct})$. [14]
     - if $\mathsf{lb} \notin L_{[1:i]}$, then $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$.

   $\mathcal{B}$ computes an FE key for $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs its guess bit β'. $\mathcal{B}$ sends β' to the kpABE challenger as its guess bit.

We observe that if the kpABE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated $\mathsf{Hybrid}_{6a,i-1}$, else $\mathsf{Hybrid}_{6a,i}$ with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_{6a,i-1}) - \Pr(\beta' = 1|\mathsf{Hybrid}_{6a,i})| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$: Observe that $\mathcal{B}$ issues a single ABE key query, which is for circuit C_L and the challenge attribute is $\mathsf{lb}_i \in L$. Hence, by the design of C_L, $C_L(\mathsf{lb}_i) = 0$ as desired. □

□

□

Claim 4.5.1. *Assume that* FE *is secure, then* $\mathsf{Hybrid}_7 \approx_c \mathsf{Hybrid}_8$.

Proof. We show that if $\mathcal{A}$ wins with non-negligible advantage ϵ in distinguishing the two hybrids, then there exists an adversary $\mathcal{B}$ against FE security with the same advantage ϵ. The steps of the reduction are similar to the proof of Claim 4.2.3, with $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \bot, \mathsf{SKE.key}_2)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f_1, m, K, 2, \bot, \mathsf{SKE.key}_2)$. Hence, here we only argue the admissibility of $\mathcal{B}$ in the FE security game.

Admissibility of $\mathcal{B}$:
Firstly, we observe that the only key queries that $\mathcal{B}$ issues to the FE challenger are for the $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ functions, where each function is defined corresponding to a key query (lb, x) by $\mathcal{A}$. Here, $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$, $\delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$ for $\mathsf{flag} = 1$. Next, we observe that for any $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ function, $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \bot, \mathsf{SKE.key}_2)$, and $\mu_1 = (\mathsf{kpABE.mpk}, f_1, m, K, 2, \bot, \mathsf{SKE.key}_2)$ the following holds true from the definition of Re-Enc and correctness of SKE decryption,

[14] In case multiple queries with $\mathsf{lb} = \mathsf{lb}_i$ are made, we need to simulte the ciphertext multiple times. In that case, we rely on multi-challenge version of Sel-INDr, which is easily seen to be equivalent with the single challenge version.

- when $\mathsf{flag} = 0$, i.e $f_0(x) = f_1(x)$, $\tilde{m}$ in $\mathsf{kpABE.ct_{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$, computed inside the $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ function is same for both f_0 and f_1, and hence same on both the inputs μ_0 and μ_1. So,

$$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) = \mathsf{kpABE.ct_{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 0).$$

$$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) = \mathsf{kpABE.ct_{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 0).$$

- When $\mathsf{flag} = 1$,i.e $f_0(x) \neq f_1(x)$, by definition of $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ we have

$$\begin{aligned}\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) &= \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \gamma_2) \\ &= \mathsf{kpABE.ct}'_{\mathsf{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 1\ ).\end{aligned}$$

$$\begin{aligned}\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) &= \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \gamma_2) \\ &= \mathsf{kpABE.ct}'_{\mathsf{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 1).\end{aligned}$$

This establishes the admissibility of $\mathcal{B}$. □

The rest of the hybrids, Hybrid_9 to Hybrid_{15}, are simply unwinding the previous hybrids and their proofs of indistinguishability are same as their corresponding counterparts in the first set of hybrids and hence, omitted. □

Message Hiding

Theorem 4.6. *Assume that F is a secure* PRF, SKE *is correct and secure,* FE *and* kpABE *are secure as per definitions 2.6 and 2.15, respectively. Furthermore, assume $|\mathcal{F}_\lambda| \leq \mathrm{poly}(\lambda)$ and $|\mathcal{M}_\lambda| \leq \mathrm{poly}(\lambda)$. Then the construction for* RPE *satisfies message hiding property as defined in Def. 3.3.*

Proof. Recall that for message hiding we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m_0, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m_1, L),$$

where for all the key queries (lb, x), either $f(x) = 0$ or $\mathsf{lb} \in L$. The proof is given via a similar sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$ as in the proof of Theorem 4.2. The hybrids are defined as follows:

Hybrid_0. This is the real world with $\beta = 0$, i.e., the challenge ciphertext is computed using the message m_0. We write the complete game here to set up notations and easy reference in later hybrids.

1. The adversary outputs the challenge messages (m_0, m_1) and the challenge function f[15].
2. The challenger generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends RPE.mpk to the adversary.
3. **Key Queries :** When adversary issues a key query on (lb, x), the challenger does the following:
   - Samples random values $\gamma_1, \gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

[15] To keep the proofs and notations simple, we let f, m_0, m_1 to be output selectively. This is sufficient to achieve security as in Def 3.3, as mentioned in Remark 3.9.

- Defines the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ as in the Figure 1.
- Computes $\mathsf{FE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta])$.
- Returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$ to the adversary.

4. **Challenge Query :** When the adversary outputs the revocation list L for the challenge ciphertext, the challenger does the following:
   - Samples a PRF key K.
   - Generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
   - Computes FE.ct as
   $$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_0, K, 0, \perp, \perp)).$$
   - Defines C_L as in Eq. 4.1 and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to the adversary.
5. In the end, the adversary outputs a bit β'.

Hybrid_1. This hybrid is same as the previous hybrid except the following changes:

- The challenger generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ in the beginning of the game after the adversary outputs (m_0, m_1) and f. It also samples a SKE secret key $\mathsf{SKE.key}_1$ and a PRF key K.
- For each key query (lb, x), γ_1 is computed differently. In particular, the challenger does the following:
  - Sets $\tilde{m}_0 = \begin{cases} m_0 & \text{if } f(x) = 1, \\ 0 & \text{if } f(x) = 0. \end{cases}$
  and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_0; F(K, (\mathsf{lb}, x)))$.
  - Sets γ_1 as $\mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hybrid_2. This hybrid is same as the previous hybrid except the following:

- The challenger samples two SKE secret keys $\mathsf{SKE.key}_1$ and $\mathsf{SKE.key}_2$.
- For each key query (lb, x), γ_2 and δ are computed differently as follows:
  - Set $\mathsf{flag} = \begin{cases} 1 & \text{if } f(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$
  - Set $\delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$, else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hybrid_3. This hybrid is same as the previous hybrid except that FE.ct in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_0, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hybrid_4. This hybrid is same as the previous hybrid except that for each key query (lb, x) $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as follows:

- If $\mathsf{flag} = 1$, $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_0; r)$, where $r \leftarrow \{0,1\}^t$,
- Else, if $\mathsf{flag} = 0$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_0; F(K, (\mathsf{lb}, x)))$. (This is same as in the previous hybrid).

Hybrid_5. This hybrid is same as the previous hybrid except that FE.ct in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2)).$$

Hybrid_6. This hybrid is same as the previous hybrid except that for all the key queries (lb, x), $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hybrid_7. This hybrid is same as the previous hybrid except that for each such key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.

Hybrid_8. This hybrid is same as the previous hybrid except that FE.ct in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2)).$$

We note that the hybrids hereafter are unwinding the changes made in the previous hybrids.

Hybrid_9. This hybrid is same as the previous hybrid except that for each key query (lb, x) with $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is changed back to $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_1; r)$, where $r \leftarrow \{0,1\}^t$.

Hybrid_{10}. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hybrid_{11}. This hybrid is same as the previous hybrid except that FE.ct in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_1, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hybrid_{12}. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_1; F(K, (\mathsf{lb}, x)))$.

Hybrid_{13}. This hybrid is same as the previous hybrid except that FE.ct in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_1, K, 0, \perp, \perp)).$$

Hybrid_{14}. This hybrid is same as the previous hybrid except that the γ_2, δ are now sampled uniformly from $\mathcal{CT}_{\mathsf{SKE}}$ for all the key queries.

Hybrid_{15}. This hybrid is same as the previous hybrid except that γ_1 is now sampled uniformly from $\mathcal{CT}_{\mathsf{SKE}}$ for all the key queries. This is the real world where the message m_1 is encrypted in the challenge ciphertext.

Indistinguishability of hybrids: The indistinguishability between the consecutive hybrids is argued in the same way as that in the proof of Theorem 4.2. Therefore, here we give only a brief sketch.

$\mathsf{Hybrid}_0 \approx_c \mathsf{Hybrid}_1 \approx_c \mathsf{Hybrid}_2$ from SKE security. $\mathsf{Hybrid}_2 \approx_c \mathsf{Hybrid}_3$ due to FE security and SKE correctness and $\mathsf{Hybrid}_3 \approx_c \mathsf{Hybrid}_4$ follows from PRF security. $\mathsf{Hybrid}_4 \approx_c \mathsf{Hybrid}_5$ follows from FE security and SKE correctness and $\mathsf{Hybrid}_5 \approx_c \mathsf{Hybrid}_6$ follows again from the SKE security. $\mathsf{Hybrid}_6 \approx_c \mathsf{Hybrid}_7$ follows from selective security of ABE. We observe that in both Hybrid_6 and Hybrid_7, $\mathsf{ABE.ct}'_{\mathsf{lb}}$ is not used when $\mathsf{flag} = 0$ and when $\mathsf{flag} = 1$, it is sampled from $\mathcal{CT}_{\mathsf{kpABE}}$ directly which can be efficiently done without using $\mathsf{kpABE.mpk}$. This lets the reduction go through. The steps of reduction are same as in the proof of Claim 4.2.7. $\mathsf{Hybrid}_7 \approx_c \mathsf{Hybrid}_8$ follows again from the security of FE and SKE correctness. In particular, we observe that here the FE challenge messages are $\mu_0 = (\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2)$. For every $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ function (corresponding to RPE key query (lb, x)) for which FE key is generated, we have the following:

- When $\mathsf{flag} = 0$, this implies $f(x) = 0$, which in turn implies that $\tilde{m_0} = \tilde{m_1} = 0$. Hence,

$$\begin{aligned}
& \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, 0; F(K, (\mathsf{lb}, x))) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1).
\end{aligned}$$

- When $\mathsf{flag} = 1$,

$$\begin{aligned}
& \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \gamma_2) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1).
\end{aligned}$$

This satisfies the admissibility condition for FE security. The rest of the hybrids undo the changes made so far to get to the real world with $\beta = 1$ and the arguments for indistinguishability are same as their counterparts in the first set of hybrids. □

5 Revocable Mixed Functional Encryption

5.1 Definition

A revocable mixed functional encryption (RMFE) scheme with input domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in [\mathbb{N}]}$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in [\mathbb{N}]}$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda \in [\mathbb{N}]}$ has the following syntax.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x) \to \mathsf{sk}_{\mathsf{lb},x}$. The key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}_\lambda$ and an input $x \in \mathcal{X}_\lambda$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},x}$.

$\mathsf{PK\text{-}Enc}(\mathsf{mpk}, L) \to \mathsf{ct}$. The public key encryption algorithm takes as input the master public key mpk and a revocation list $L \subseteq \mathcal{L}_\lambda$ and outputs a ciphertext ct.

$\mathsf{SK\text{-}Enc}(\mathsf{msk}, f, L) \to \mathsf{ct}$. The secret key encryption algorithm takes as input the master secret key msk, a function $f \in \mathcal{F}_\lambda$ and a revocation list $L \subseteq \mathcal{L}_\lambda$, and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, L, \mathsf{ct}) \to \{0,1\}$. The decryption algorithm takes the secret key $\mathsf{sk}_{\mathsf{lb},x}$, a revocation list $L \subseteq \mathcal{L}_\lambda$ and a ciphertext ct and outputs a bit.

Definition 5.1 (Correctness)**.** A RMFE scheme is said to be correct if there exists negligible functions $\mathsf{negl}_1(\cdot), \mathsf{negl}_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\ \mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, L, \mathsf{ct}) = 1 : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{sk}_{\mathsf{lb},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x); \\ \mathsf{ct} \leftarrow \mathsf{PK\text{-}Enc}(\mathsf{mpk}, L) \end{array}\ \right] \geq 1 - \mathsf{negl}_1(\lambda).$$

$$\mathsf{lb} \notin L \Rightarrow \Pr\left[\ \mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, L, \mathsf{ct}) = f(x) : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{sk}_{\mathsf{lb},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x); \\ \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f, L) \end{array}\ \right] \geq 1 - \mathsf{negl}_2(\lambda).$$

Security. Here we define the security requirements of RMFE scheme.

Definition 5.2 (q-query Mode Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RMFE scheme satisfies q-query mode hiding security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\ \beta' = \beta : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ f, L \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot), \mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_0 \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f, L); \\ \mathsf{ct}_1 \leftarrow \mathsf{PK\text{-}Enc}(\mathsf{mpk}, L); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot), \mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\ \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the $\mathsf{SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle and is admissible only if for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, $f(x) = 1$.

Definition 5.3 (q-query Selective Function Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RMFE scheme satisfies q-query selective function hiding security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\ \beta' = \beta : \begin{array}{l} L \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (f_0, f_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot), \mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f_\beta, L); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot), \mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\ \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the $\mathsf{SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle and for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

Remark 5.4. We note that when the function space $\mathcal{F}_\lambda$ is polynomially small and q is a constant, a variant of Definition 5.3 where the adversary outputs the challenge functions (f_0, f_1) and the $\mathsf{SK\text{-}Enc}$ query functions $\{\bar{f}_i\}_{i \in [q]}$ at the beginning of the game, before the $\mathsf{Setup}(1^\lambda)$ algorithm is run, is equivalent to Definition 5.3 where the adversary adaptively outputs the challenge functions (f_0, f_1) and can make $\mathsf{SK\text{-}Enc}$ queries adaptively, with polynomial loss. Similar comment also applies to Definition 5.2. We will use these simplifications in the security proofs.

5.2 Construction

In this section we give a construction of 1-query secure RMFE scheme, with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \rightarrow \{0,1\}\}$ and a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_\lambda$. We assume that the size of $|\mathcal{F}_\lambda|$ is bounded by some polynomial in λ, which will suffice for our purpose.
Our scheme uses the following building blocks:

1. A 2-bounded semi-adaptive simulation based function-message private (Definition 2.11) SKFE scheme $\mathsf{SKFE} = (\mathsf{SKFE.Setup}, \mathsf{SKFE.KeyGen}, \mathsf{SKFE.Enc}, \mathsf{SKFE.Dec})$ that supports the function class $\mathcal{F}$. This can be instantiated from one-way functions (Lemma 2.12).

2. A key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.Enc}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Dec})$ for the circuit class $\mathcal{C}_{\ell(\lambda),d(\lambda)}$ with message space $\{0,1\}^\lambda$ satisfying Sel-IND security (Definition 2.14) and efficiency properties described in Theorem 2.17. We set $\ell(\lambda) = \ell_{\mathsf{lb}} + \log(\lambda) + 1$ and $d(\lambda) = \omega(\log \lambda)$, where ℓ_{lb} is the label length.[16] This can be instantiated from the LWE assumption (Theorem 2.17).

3. A lockable obfuscation scheme $\mathsf{LO} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$ with lock space $\{0,1\}^\lambda$ that supports circuits of the form CC defined in Fig. 2. As we will analyze later, the circuit is of fixed polynomial size in λ and $|f|$, where $|f|$ is the description size of the function $f \in \mathcal{F}$. This can be instantiated from the LWE assumption (Theorem. 2.24).

Below we describe our construction of a 1-query secure RMFE scheme $\mathsf{RMFE} = (\mathsf{RMFE.Setup}, \mathsf{RMFE.KeyGen}, \mathsf{RMFE.PK\text{-}Enc}, \mathsf{RMFE.SK\text{-}Enc}, \mathsf{RMFE.Dec})$.

$\mathsf{RMFE.Setup}(1^\lambda) \to (\mathsf{RMFE.mpk}, \mathsf{RMFE.msk})$. The setup algorithm does the following:

- Generate $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$.
- Generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
- Output $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.

$\mathsf{RMFE.KeyGen}(\mathsf{RMFE.msk}, \mathsf{lb}, x) \to \mathsf{RMFE.sk}_{\mathsf{lb},x}$. The key generation algorithm does the following:

- Parse $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.
- For all $j \in [\lambda], b \in \{0,1\}$, sample $K_{j,b}, R_{j,b} \leftarrow \{0,1\}^\lambda$.
  Denote $K = \{K_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ and $R = \{R_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$.
- Compute
$$\mathsf{SKFE.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, (x, K, R)).$$
- For all $j \in [\lambda], b \in \{0,1\}$, compute
$$\mathsf{kpABE.ct}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), K_{j,b}).$$
- Output $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$.

$\mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L) \to \mathsf{RMFE.ct}$. The public key encryption algorithm does the following:

- Computes a simulated code $\mathsf{RMFE.ct} \leftarrow \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})$[17].
- It outputs $\mathsf{RMFE.ct}$ as the ciphertext.

$\mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L) \to \mathsf{RMFE.ct}$. The secret key encryption algorithm does the following:

- Parse $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$, and sample a tag $\mathbf{z} \leftarrow \{0,1\}^\lambda$ and a lock value $\alpha \leftarrow \{0,1\}^\lambda$.

[16] Concretely, we can choose $d(\lambda) = \Theta(\log \lambda \log \log \lambda)$ for example.
[17] Here, CC represents the maximum possible size of $\mathsf{CC}[\cdot,\cdot]$ circuit defined in Figure 2.

- For all $j \in [\lambda]$, compute $\mathsf{kpABE.sk}_{L,j,z_j} \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_{L,j,z_j})$, where the function C_{L,j,z_j} has L, j and z_j hardwired and is defined as follows :
  On input $(\mathsf{lb}, i, b) \in \mathcal{L}_\lambda \times [\lambda] \times \{0,1\}$,

$$C_{L,j,z_j}(\mathsf{lb}, i, b) = \begin{cases} 1 & \text{if } (\mathsf{lb} \notin L) \wedge (i = j) \wedge (b = z_j) \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

- Compute $\mathsf{SKFE.sk} \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f,\mathbf{z},\alpha})$, where the function $P_{f,\mathbf{z},\alpha}$ has $f, \mathbf{z}, \alpha$ hardwired and is defined as follows :
  On input $x \in \mathcal{X}_\lambda$, $K = \{K_{j,b}\}_{j \in [\lambda], b \in \{0,1\}}$, $R = \{R_{j,b}\}_{j \in [\lambda], b \in \{0,1\}}$,

$$P_{f,\mathbf{z},\alpha}(x, K, R) = \begin{cases} \bigoplus_j K_{j,z_j} \oplus \alpha & \text{if } f(x) = 0 \\ \bigoplus_j R_{j,z_j} & \text{if } f(x) = 1. \end{cases} \tag{5.2}$$

- Construct function $\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j \in [\lambda]}]$, with $\mathsf{SKFE.sk}$ and $\{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j \in [\lambda]}$ hardwired and is defined as in Figure 2.
- Output $\mathsf{RMFE.ct} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j \in [\lambda]}], \alpha)$.

$\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) \to \{0,1\}$. The decryption algorithm does the following:

- Parse $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j \in [\lambda], b \in \{0,1\}})$ and $\mathsf{RMFE.ct} = \widetilde{\mathsf{CC}}$, where $\widetilde{\mathsf{CC}}$ is regarded as an obfuscated circuit of LO.
- For all $j \in [\lambda], b \in \{0,1\}$, compute

$$\mathsf{kpABE.off}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Dec}^{\mathsf{off}}(\mathsf{kpABE.mpk}, C_{L,j,b}, (\mathsf{lb}, j, b)).$$

- Compute

$$y \quad = \quad \mathsf{LO.Eval}\big(\widetilde{\mathsf{CC}}, (\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j \in [\lambda], b \in \{0,1\}})\big).$$

- Output 1 if $y = \perp$, else output 0.

Remark 5.5. We note that by performing the part of the ABE decryption that uses $C_{L,j,b}$, outside of CC , we do not need to provide $C_{L,j,b}$ (or L) as input to CC. Instead, we provide $\{\mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j \in [\lambda], b \in \{0,1\}}$ whose size is independent of the size of $C_{L,j,b}$ (and thus that of L). This helps us in getting succinct ciphertext.

Correctness. We prove the correctness via the following theorem.

Theorem 5.6. *Suppose* kpABE, LO *and* SKFE *are correct and* LO *is secure, then the above construction of* RMFE *satisfies correctness as defined in Def. 5.1.*

Proof. We consider the following two cases:

1. **Public Encryption Correctness.**
   For $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$, we have that $\mathsf{RMFE.ct} = \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})$. Firstly, we note that from LO security, RMFE.ct is indistinguishable from $\mathsf{RMFE.ct}'$ computed as $\mathsf{LO.Obf}(C, \alpha)$, for any circuit C of the same size as that used by the simulator and has output of length λ. Now, since $\alpha \leftarrow \{0,1\}^\lambda$ has high entropy, for all but negligible inputs w, $C(w) \neq \alpha$. So, from the correctness of LO, it follows that with all but negligible probability

$$\mathsf{LO.Eval}(\mathsf{RMFE.ct}, (\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j \in [\lambda], b \in \{0,1\}})) \quad = \quad \perp.$$

   Hence the RMFE.Dec algorithm outputs 1 with all but negligible probability.

Function $\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}]$

Hardwired values: A SKFE secret key SKFE.sk and kpABE keys $\{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}$.
Inputs: A SKFE ciphertext SKFE.ct and kpABE ciphertexts $\{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$.
Output : A binary string $\alpha^* \in \{0,1\}^\lambda$.

1. For all $j \in [\lambda]$, compute $m_j = \mathsf{kpABE.Dec}^{\mathsf{on}}(\mathsf{kpABE.sk}_{L,j,z_j}, \mathsf{kpABE.ct}_{\mathsf{lb},j,z_j}, \mathsf{kpABE.off}_{\mathsf{lb},j,z_j})$.
   Let $M_0 = \bigoplus_j m_j$
2. Compute $M_1 = \mathsf{SKFE.Dec}(\mathsf{SKFE.sk}, \mathsf{SKFE.ct})$.
3. Output $M_1 \oplus M_0$.

Figure 2: Compute and Compare function CC

2. **Secret Encryption Correctness.**
   For $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$, we have $\mathsf{RMFE.ct} = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}], \alpha)$. Now consider the following steps of
   $\mathsf{LO.Eval}(\mathsf{RMFE.ct}, (\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}))$

   - Since $\mathsf{lb} \notin L$, by correctness of kpABE, for all $j \in [\lambda]$, $M_0 = \oplus_j m_j = \oplus_j K_{j,z_j}$ with all but negligible probability.
   - By correctness of SKFE, if $f(x) = 0$, we have $M_1 = \bigoplus_j K_{j,z_j} \oplus \alpha$, else $M_1 = \bigoplus_j R_{j,z_j}$.
   - We have $M_0 \oplus M_1 = \alpha$ if $f(x) = 0$.

   So, by the correctness of LO, LO.Eval outputs 1 if $f(x) = 0$, $\perp$ otherwise with all but negligible probability. Hence the RMFE.Dec algorithm, by construction, outputs 0 if $f(x) = 0$ and 1 if $f(x) = 1$ with all but negligible probability.

This proves the correctness of the above construction. □

Efficiency. Here we argue that our construction achieves optimal parameters. Namely, we show that the sizes of the parameters are independent of $|L|$. We first observe that $C_{L,j,b}$ as defined in Eq (5.1) can be implemented with depth $d = \omega(\log \lambda)$, since the membership check $\mathsf{lb} \overset{?}{\in} L$ can be done with depth $\log(|\mathsf{lb}| \cdot |L|) = \log(\mathrm{poly}(\lambda))$ and the equality check can be done with depth $\log(|j|) = \log\log\lambda$. We then bound the size of parameters.

1. Public key size $|\mathsf{RMFE.mpk}|$: By the efficiency property of kpABE (Theorem 2.17), we have $|\mathsf{RMFE.mpk}| = |\mathsf{kpABE.mpk}| = \mathrm{poly}(\lambda, d, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{lb}|)$.
2. Secret key size $|\mathsf{RMFE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RMFE.sk}_{\mathsf{lb},x}| = |\mathsf{SKFE.ct}| + |\mathsf{kpABE.mpk}| + 2 \cdot \lambda(|(\mathsf{lb}, j, b)| + |\mathsf{kpABE.ct}|)$. The first term can be bounded by $\mathrm{poly}(\lambda, |f|, |x|)$, the second is $\mathrm{poly}(\lambda, |\mathsf{lb}|)$, and the last terms is $\mathrm{poly}(\lambda, |\mathsf{lb}|)$. Therefore, the total size is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

3. Ciphertext size $|\mathsf{RMFE.ct}|$: We first bound the size of the circuit CC defined in Fig. 2. The dominant operations in the circuit is the decryption of SKFE and the online decryption of kpABE. We can see that the former is implemented by a circuit of size $\mathrm{poly}(\lambda, |x|, |f|)$ by the efficiency of SKFE. The latter can be implemented by a circuit of size $\mathrm{poly}(\lambda, \ell, d) = \mathrm{poly}(\lambda, \mathsf{lb})$ by the online efficiency of kpABE (Theorem 2.17). Therefore, the total size CC is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$. By the efficiency of LO, the size of the ciphertext is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ as well.

4. Depth of the circuit implementing RMFE.Dec: We also evaluate the depth of the circuit implementing RMFE.Dec and show that it is independent from $|L|$, since it will be used later in Sec. 6. We first observe that the dominant operations in RMFE.Dec are the offline decryption of kpABE and the evaluation of CC. The depth of the former can be bounded by $\mathrm{poly}(\lambda, \ell, \mathsf{depth}(C_{L,j,b})) \leq \mathrm{poly}(\lambda, |\mathsf{lb}|)$ by Theorem 2.17. The depth of the latter can be bounded by its size and thus is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ as we have seen in the previous item. The total depth is thus bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$, which is independent from $|L|$.

5.3 Security

In this section we show that our construction of RMFE scheme satisfies all the security requirements.
We will use the following notations in the security proof:
For $X \in \{K, R\}$,

- $X := \{X_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$, $X_b := \{X_{j,b}\}_{j\in[\lambda]}$, $X_j := \{X_{j,b}\}_{b\in\{0,1\}}$.
- For any vector $\mathbf{z} \in \{0,1\}^\lambda$, $X_{\mathbf{z}} := \{X_{j,z_j}\}_{j\in[\lambda]}$.

Mode hiding

Theorem 5.7. *Assume that* SKFE *and* LO *are secure as per definitions 2.11 and 2.23, respectively. Furthermore, assume $|\mathcal{F}_\lambda| \leq \mathrm{poly}(\lambda)$. Then the* RMFE *construction satisfies 1-query mode hiding security as per Definition 5.2.*

Proof. Recall that for mode hiding, we need

$$\mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f^*, L^*) \approx_c \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L^*),$$

where for all key queries (lb, x), $f^*(x) = 1$.
The proof proceeds via the following sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$.

Hybrid_0 : This is the real world with $\beta = 0$, where the challenge ciphertext for (f^*, L^*) is computed using the RMFE.SK-Enc algorithm. We write the complete game here to set up the notations and easy reference in later hybrids.

1. The adversary outputs the challenge function f^* and the SK-Enc query function $\bar{f}$[18].
2. The challenger generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$. It sends RMFE.mpk to $\mathcal{A}$.

[18]To keep the proofs and notations simple, we let f^* and $\bar{f}$ to be given selectively. This is sufficient to achieve security as in Def 5.2, as mentioned in Remark 5.4.

3. **Key Queries:** For each key query (lb, x), the challenger computes $\mathsf{SKFE.ct}$ and $\mathsf{kpABE.ct}_{\mathsf{lb},j,b}$ as in the construction and returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$ to $\mathcal{A}$.
4. **Challenge Query:** When the adversary outputs L^* for the challenge query, the challenger does the following:
   - Samples a tag $\mathbf{z}^* \leftarrow \{0,1\}^\lambda$ and a lock value $\alpha^* \leftarrow \{0,1\}^\lambda$.
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$ and a SKFE secret key $\mathsf{SKFE.sk}^*$ as in the construction.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}]$ as defined in Figure 2 and returns $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ to the adversary $\mathcal{A}$.
5. SK-Enc **Query:** When the adversary outputs $\bar{L}$ for the SK-Enc query, the challenger does the following:
   - Samples a tag $\bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$ and a lock value $\bar{\alpha} \leftarrow \{0,1\}^\lambda$.
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$ and a SKFE secret key $\mathsf{SKFE.}\bar{\mathsf{sk}}$ as in the construction.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}]$ and returns $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$ to the adversary $\mathcal{A}$.
6. In the end, $\mathcal{A}$ outputs a bit β'.

Hybrid_1 : This hybrid is same as the previous hybrid except the following changes:

1. The challenger samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}}$ in the beginning of the game after the adversary outputs $f^*, \bar{f}$.
2. The challenger then computes $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f^*,\mathbf{z}^*,\alpha^*}|})$ and $\mathsf{SKFE.}\bar{\mathsf{sk}} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{\mathrm{poly}(|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$ in this order using SKFE simulators.

3. The key generation, challenge and SK-Enc queries are answered as follows:
   - For each key query (lb, x), the SKFE ciphertext in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, \bigoplus_j R_{j,z_j^*}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
     Note that $P_{f^*,\mathbf{z}^*,\alpha^*}(x, K, R) = \bigoplus_j R_{j,z_j^*}$, due to admissibility requirement.
   - To answer the challenge and the SK-Enc queries, $\mathsf{SKFE.sk}^*$ and $\mathsf{SKFE.}\bar{\mathsf{sk}}$ generated by SKFE simulators in Step 2 are used for generating $\mathsf{RMFE.ct}^*$ and $\mathsf{RMFE.}\bar{\mathsf{ct}}$, respectively.

Hybrid_2 : This hybrid is same as the previous hybrid except that the challenger uses $\mathsf{LO.Sim}$ to generate the challenge ciphertext.

$$\mathsf{RMFE.ct}^* = \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})^{19}.$$

Hybrid_3 : This hybrid is same as the previous hybrid except that the challenger uses $\mathsf{SKFE.Enc}$ and $\mathsf{SKFE.KeyGen}$ to generate SKFE ciphertexts and keys respectively. Formally,

$$\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}})$$

[19] Here, $|\mathsf{CC}|$ represents the maximum size of the circuit $\mathsf{CC}[\cdot,\cdot]$ defined in Figure 2.

For each key query (lb, x),

$$\mathsf{SKFE.ct} = \mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, (K, R, x)),$$

where vectors K and R are freshly sampled for each key as defined in the construction. This is the real world with $\beta = 1$, where the challenge ciphertext is computed using the RMFE.PK-Enc algorithm.

Indistinguishability of hybrids

Claim 5.7.1. *Assume that* SKFE *is secure (Def. 2.11), then* $\mathsf{Hybrid}_0 \approx_c \mathsf{Hybrid}_1$.

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ who can distinguish between Hybrid_0 and Hybrid_1 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of SKFE scheme with the same advantage ϵ. The reduction is as follows.

1. Upon being invoked by the SKFE challenger, $\mathcal{B}$ invokes $\mathcal{A}$ which outputs $f^*, \bar{f}$.
2. $\mathcal{B}$ samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$.
3. $\mathcal{B}$ defines the functions $P_{f^*,\mathbf{z}^*,\alpha^*}$ and $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$ as defined in Eq. (5.2) and sends it to the SKFE challenger in this order as key queries. The challenger generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$ and samples $\hat{\beta} \leftarrow \{0,1\}$.
   If $\hat{\beta} = 0$, it computes $\mathsf{SKFE.sk}^* = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f^*,\mathbf{z}^*,\alpha^*})$ and $\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}})$
   else it computes $\mathsf{SKFE.sk}^* = \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f^*,\mathbf{z}^*,\alpha^*}|})$, $\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$ and sends $\{\mathsf{SKFE.sk}^*, \mathsf{SKFE.}\bar{\mathsf{sk}}\}$ to $\mathcal{B}$.
4. $\mathcal{B}$ generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and sends RMFE.mpk to $\mathcal{A}$.
5. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:
   - Samples $K_{j,b}, R_{j,b} \leftarrow \{0,1\}^\lambda$, $\forall j \in [\lambda], b \in \{0,1\}$.
   - Computes $\mathsf{kpABE.ct}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.msk}, (\mathsf{lb}, j, b), K_{j,b})$ for $j \in [\lambda], b \in \{0,1\}$.
   - It sends (x, K, R) as challenge message to the SKFE challenger. The challenger returns $\mathsf{SKFE.ct}_{\hat{\beta}}$, where $\mathsf{SKFE.ct}_0 = \mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, (x, K, R))$ and $\mathsf{SKFE.ct}_1 = \mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, \bigoplus_j R_{j,z_j^*}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
   - Returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}_{\hat{\beta}}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j,b})$ to $\mathcal{A}$.
6. **Challenge Query:** When $\mathcal{A}$ outputs L^* for the challenge query, $\mathcal{B}$ does the following:
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}]$ as defined in Figure 2 and returns $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ to $\mathcal{A}$.
7. SK-Enc **Query:** When the adversary outputs $\bar{L}$ for the SK-Enc query, $\mathcal{B}$ does the following:
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}]$ and returns $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$ to $\mathcal{A}$.

8. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards β' as its guess bit to the SKFE challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hybrid_0, else Hybrid_1 with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_0) - \Pr(\beta' = 1|\mathsf{Hybrid}_1)|$ = ϵ (by assumption). ☐

Claim 5.7.2. *Assume that* LO *is secure (Def. 2.23), then* $\mathsf{Hybrid}_1 \approx_c \mathsf{Hybrid}_2$.

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ who can distinguish between Hybrid_1 and Hybrid_2 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of LO scheme with the same advantage ϵ. The reduction is as follows

1. Upon being invoked by the LO challenger, $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs $f^*, \bar{f}$.

2. $\mathcal{B}$ samples $\bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$. It also defines the function $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$.

3. $\mathcal{B}$ generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$ and sends $\mathsf{RMFE.mpk}$ to $\mathcal{A}$.

4. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:

   - Samples $K_{j,b}, R_{j,b} \leftarrow \{0,1\}^\lambda$, $\forall j \in [\lambda], b \in \{0,1\}$.
   - Computes $\mathsf{kpABE.ct}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), K_{j,b})\ \forall j \in [\lambda], b \in \{0,1\}$ and $\mathsf{SKFE.ct} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, \bigoplus_j R_{j,z_j^*}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
   - Returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j,b})$ to $\mathcal{A}$.

5. **Challenge Query:** When the adversary outputs L^* for the challenge ciphertext , $\mathcal{B}$ does the following:

   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$.
   - Constructs a function $\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}]$ as defined in Figure 2, where $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f^*,\mathbf{z}^*,\alpha^*}|})$[20].
   - It sends $\mathsf{CC}[\mathsf{SKFE.sk}^*, \mathsf{kpABE.sk}_{L^*,j,z_j^*}]$ to the LO challenger. The challenger samples a lock value $\alpha^* \leftarrow \{0,1\}^\lambda$ and $\hat{\beta} \leftarrow \{0,1\}$, computes and return $\mathsf{Obf}_{\hat{\beta}}$, where $\mathsf{Obf}_0 = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ and $\mathsf{Obf}_1 = \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})$.
   - It sets $\mathsf{RMFE.ct}^* = \mathsf{Obf}_{\hat{\beta}}$ and sends it to the adversary $\mathcal{A}$.

6. SK-Enc **Query:** When the adversary outputs $\bar{L}$ for the SK-Enc query, $\mathcal{B}$ does the following:

   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}]$, where $\mathsf{SKFE.}\bar{\mathsf{sk}} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$.
   - Computes $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}}\}_{j\in[\lambda]}], \bar{\alpha})$ and returns it to the adversary $\mathcal{A}$.

7. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards β' as its guess bit to the LO challenger.

[20]The size of $P_{f^*,\mathbf{z}^*,\alpha^*}$ is independent of any specific lock value and hence can be computed without the knowledge of α^*.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hybrid_1, else Hybrid_2 with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_1) - \Pr(\beta' = 1|\mathsf{Hybrid}_2)|$ = ϵ (by assumption). □

Claim 5.7.3. *Assume that* SKFE *is secure (Def. 2.11), then* $\mathsf{Hybrid}_2 \approx_c \mathsf{Hybrid}_3$.

The proof of this claim is similar to that of Claim 5.7.1, hence omitted. □

Function Hiding

Theorem 5.8. *Assume* SKFE *is secure (Def. 2.11),* kpABE *satisfies* Sel-IND *security (Def. 2.14). Furthermore, assume* $|\mathcal{F}_\lambda| \leq \mathrm{poly}(\lambda)$*. Then, the* RMFE *construction satisfies* 1*-query function hiding as defined in Definition 5.3.*

Proof. We prove the theorem via the following sequence of hybrids.

Hybrid_0 : This is the real world with $\beta = 0$. We summarize the steps of the game here to set up the notations used in the later hybrids.

1. The adversary outputs the challenge query f_0, f_1, L^* and the SK-Enc query function $\bar{f}$ in the beginning of the game[21]. The challenger then does the following:
   - Generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.
   - It also samples a tag $\mathbf{z}^* \in \{0,1\}^\lambda$ and a lock value α^*.
   - Computes $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*})$ and $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$ as defined in the construction.
   - Returns $\mathsf{RMFE.mpk}$ and the challenge ciphertext $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ to $\mathcal{A}$.
2. **Key Queries:** For each key query (lb, x), the challenger returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$, where $\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ are computed as in the construction.
3. SK-Enc **Query :** When the adversary outputs $\bar{L}$ for the SK-Enc query, the challenger does the following:
   - Samples a tag $\bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$ and a lock value $\bar{\alpha} \leftarrow \{0,1\}^\lambda$.
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$ and a SKFE secret key $\mathsf{SKFE.}\bar{\mathsf{sk}}$ as in the construction.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}]$ and returns $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$ to $\mathcal{A}$.
4. In the end, $\mathcal{A}$ outputs a bit β'.

Hybrid_1 : This hybrid is same as the previous hybrid except the following:

1. The challenger samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}}$ and defines the functions $P_{f_0,\mathbf{z}^*,\alpha^*}$ and $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$ in the beginning of the game.
2. The SKFE ciphertexts and keys are computed using SKFE simulators as:

[21] To keep the proofs and notations simple, we let f_0, f_1 and $\bar{f}$ to be given selectively. This is sufficient to achieve security as in Def 5.2, as mentioned in Remark 5.4.

- For each key query (lb, x), the SKFE ciphertext in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*}(x, K, R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
- The SKFE secret keys $\mathsf{SKFE.sk}^*$ and $\mathsf{SKFE.}\bar{\mathsf{sk}}$ in $\mathsf{RMFE.ct}^*$ and $\mathsf{RMFE.}\bar{\mathsf{ct}}$ are computed as $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_0,\mathbf{z}^*,\alpha^*}|})$ and $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$, respectively.

Hybrid_2 : In this hybrid, for each key query (lb, x) with $\mathsf{lb} \in L^*$, for all $j \in [\lambda]$, $\mathsf{kpABE.ct}_{\mathsf{lb},j,1-\bar{z}_j}$ in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, 1 - \bar{z}_j), 0^\lambda)$.

Hybrid_3 : In this hybrid, for each key query (lb, x) with $\mathsf{lb} \in L^*$, K-values are chosen differently as follows: let $i \in [\lambda]$ be the first position where $z_i^* \neq \bar{z}_i$. If no such i exists, then the challenger aborts the game. Else, it samples $\{K_{j,b}\}_{j\in[\lambda]\setminus\{i\},b\in\{0,1\}}$, $K_{i,\bar{z}_i}$, $\{R_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ uniformly randomly as in the original game. It then sets $K_{i,1-\bar{z}_i} = \bigoplus_{j\in[\lambda]\setminus\{i\}} K_{j,z_j^*} \oplus \alpha^* \oplus \bigoplus_{j\in[\lambda]} R_{j,z_j^*}$.

Hybrid_4 : This hybrid is same as the previous hybrid, except that the SKFE ciphertexts and keys in RMFE secret keys and ciphertexts are now generated with function f_1 in place of f_0 as follows:

- For each key query (lb, x), the SKFE ciphertext in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_1,\mathbf{z}^*,\alpha^*}(x, K, R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
- The SKFE secret key $\mathsf{SKFE.sk}^*$ in $\mathsf{RMFE.ct}^*$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_1,\mathbf{z}^*,\alpha^*}|})$.

Now, from here we undo the changes made in the previous hybrids.

Hybrid_5 : In this hybrid, for each key query (lb, x) with $\mathsf{lb} \in L^*$, $K_{i,1-\bar{z}_i}$ (where i is the first position where $\mathbf{z}^*$ and $\bar{\mathbf{z}}$ differ) is also sampled randomly.

Hybrid_6 : In this hybrid, for each such key query (lb, x) with $\mathsf{lb} \in L^*$, $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, 1-\bar{z}_j), 0^\lambda)$ is changed back to $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, 1-\bar{z}_j), K_{j,1-\bar{z}_j})$ in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$.

Hybrid_7 : In this hybrid SKFE ciphertexts in RMFE secret keys and SKFE secret keys in RMFE ciphertexts are computed as in the real world. That is,

- For each key query (lb, x), $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, K, R), \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), K_{j,b})\}_{j\in[\lambda],b\in\{0,1\}})$.
- $\mathsf{RMFE.ct}^* = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$, $\mathsf{RMFE.}\bar{\mathsf{ct}} = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$, where $\mathsf{SKFE.sk}^* = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f_1,\mathbf{z}^*,\alpha^*})$ and $\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{\bar{f},\mathbf{z}^*,\alpha^*})$

Note that this hybrid corresponds to the real world for $\beta = 1$.

Indistinguishability of the hybrids. Now we show that the consecutive hybrids are computationally indistinguishable for any PPT adversary $\mathcal{A}$.

Claim 5.8.1. *Assume that* SKFE *is secure (Def. 2.11). Then,* $\mathsf{Hybrid}_0 \approx_c \mathsf{Hybrid}_1$.

Proof. The proof follows similar steps as the proof for Claim 5.7.1, hence omitted. $\square$

Claim 5.8.2. *Assume that* kpABE *is selectively secure. Then,* $\mathsf{Hybrid}_1 \approx_c \mathsf{Hybrid}_2$.

Proof. To prove the claim we consider the following sub hybrids between Hybrid_1 and Hybrid_2. Let $L^* = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_{|L^*|}\}$ with some fixed ordering between the labels in L^*. Let $L_{[1:k]} \subseteq L^*$ denote the set of first k labels in L^*, i.e. $L^*_{[1:k]} = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_k\}$. Then for all $1 \le i \le |L^*|$ and $0 \le \tau \le \lambda$, define $\mathsf{Hybrid}_{1.(i,\tau)}$: which is same as Hybrid_1 except that for any key query (lb, x), $\{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ is computed differently as follows:

$$\mathsf{kpABE.ct}_{\mathsf{lb},j,b} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), W),$$

where

$$W = \begin{cases} K_{i,b} & \text{if } (b = \bar{\mathbf{z}}_j) \vee (\mathsf{lb} \notin L_{[1:i]}) \vee (\mathsf{lb} = \mathsf{lb}_i \wedge j > \tau), \\ 0^\lambda & \text{otherwise, i.e. } (b = 1 - \bar{\mathbf{z}}_j) \wedge (\mathsf{lb} \in L_{[1:i-1]}) \vee (\mathsf{lb} = \mathsf{lb}_i \wedge j \le \tau). \end{cases}$$

Then, we observe that $\mathsf{Hybrid}_{1.(1,0)} = \mathsf{Hybrid}_1$, $\mathsf{Hybrid}_{1.(|L^*|,\lambda)} = \mathsf{Hybrid}_2$ and $\mathsf{Hybrid}_{1.(i-1,\lambda)} = \mathsf{Hybrid}_{1,(i,0)}$.

Hence, all we need to show is that for all $i \in [|L^*|]$, $\tau \in [\lambda]$,

$$\mathsf{Hybrid}_{1.(i,\tau-1)} \approx_c \mathsf{Hybrid}_{1.(i,\tau)}.$$

This follows from the Sel-IND security of kpABE. In particular, if there is no key query issued for lb_i, then the hybrids are identical. On the other hand, if there is a key query (lb, x), such that $\mathsf{lb} = \mathsf{lb}_i$, then we show that if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage ϵ then we can design a PPT algorithm $\mathcal{B}$ with the same advantage ϵ against Sel-IND security of kpABE. $\mathcal{B}$ is defined as follows:

1. Upon being invoked by the kpABE challenger, $\mathcal{B}$ invokes $\mathcal{A}$ which outputs $f_0, f_1, L^*, \bar{f}$. Let $L^* = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_{|L^*|}\}$.

2. $\mathcal{B}$ sends $(\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau)$ as challenge attribute to the kpABE challenger. The kpABE challenger samples $\hat{\beta} \leftarrow \{0,1\}$ and $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and sends kpABE.mpk to $\mathcal{B}$.

3. $\mathcal{B}$ generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$ and sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$. It also samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}}$ and defines $P_{f_0,\mathbf{z}^*,\alpha^*}$ and $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$. It then does the following:
   - For each $j \in [\lambda]$, defines the circuit $C_{L^*,j,\mathbf{z}^*_j}$ and sends a key query for $C_{L^*,j,\mathbf{z}^*_j}$ to the kpABE challenger. The kpABE challenger returns $\mathsf{kpABE.sk}_{L^*,j,\mathbf{z}^*_j}$.
   - Computes $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_0,\mathbf{z}^*,\alpha^*}|})$.
   - Computes $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,\mathbf{z}^*_j}\}_{j\in[\lambda]}])$.
   - Returns RMFE.mpk and $\mathsf{RMFE.ct}^*$ to $\mathcal{A}$.

4. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:
   - Computes $\mathsf{SKFE.ct} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*}(x, K, R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
   - To compute kpABE ciphertext, $\mathsf{kpABE.sk}_{\mathsf{lb},j,b}$ for $j \in [\lambda], b \in \{0,1\}$,
     - if $(\mathsf{lb} = \mathsf{lb}_i) \wedge j = \tau \wedge b = 1 - \bar{\mathbf{z}}_j$[22], then $\mathcal{B}$ sends challenge query with messages $\mu_0 = K_{j,1-\bar{\mathbf{z}}_j}$ and $\mu_1 = 0^\lambda$. The kpABE challenger returns $\mathsf{kpABE.ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau), \mu_{\hat{\beta}})$, which $\mathcal{B}$ sets as $\mathsf{kpABE.ct}_{\mathsf{lb}_i,\tau,1-\bar{\mathbf{z}}_\tau}$.

[22]If there are more than one key queries with $\mathsf{lb} = \mathsf{lb}_i$, then we use multi-challenge version of Sel-IND security of kpABE which can easily be shown equivalent to the one defined in Definition 2.14.

- else, $\mathcal{B}$ computes $\mathsf{kpABE.ct}_{\mathsf{lb},j,b}$ on its own using $\mathsf{kpABE.mpk}$ as defined for $\mathsf{Hybrid}_{1.(i,\tau-1)}$ (same for $\mathsf{Hybrid}_{1.(i,\tau)}$).

- Returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j,b})$ to $\mathcal{A}$.

5. SK-Enc **Query:** When $\mathcal{A}$ outputs $\bar{L}$ as part of SK-Enc query, $\mathcal{B}$ does the following:
   - Computes $\mathsf{SKFE.}\bar{\mathsf{sk}} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$.
   - For each $j \in [\lambda]$, defines circuit $C_{\bar{L},j,\bar{\mathbf{z}}_j}$ and sends a kpABE key query for $C_{\bar{L},j,\bar{\mathbf{z}}_j}$. The kpABE challenger returns $\mathsf{kpABE.sk}_{\bar{L},j,\bar{\mathbf{z}}_j}$.
   - Computes $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{\mathbf{z}}_j}\}_{j\in[\lambda]}])$.
   - Returns $\mathsf{RMFE.}\bar{\mathsf{ct}}$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards β' as its guess bit to the kpABE challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated $\mathsf{Hybrid}_{1.(i,\tau-1)}$, else $\mathsf{Hybrid}_{1.(i,\tau)}$ with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)|$ = $|\Pr(\beta' = 1|\mathsf{Hybrid}_{1.(i,\tau-1)}) - \Pr(\beta' = 1|\mathsf{Hybrid}_{1.(i,\tau)})|$ = ϵ (by assumption).

Admissibility of $\mathcal{B}$: Firstly, we observe that $\mathcal{B}$ issues key queries for only the following set of circuits: $\{C_{L^*,j,\mathbf{z}^*_j}\}_{j\in[\lambda]}$, $\{C_{\bar{L},j,\bar{\mathbf{z}}_j}\}_{j\in[\lambda]}$ and the challenge attribute is $(\mathsf{lb}_i, \tau, 1-\bar{\mathbf{z}}_\tau)$, where $\mathsf{lb}_i \in L^*$. Next, we note that

- $C_{L^*,j,\mathbf{z}^*_j}(\mathsf{lb}_i, \tau, 1-\bar{\mathbf{z}}_\tau) = 0$ for all $j \in [\lambda]$, because $\mathsf{lb}_i \in L^*$.
- $C_{\bar{L},j,\bar{\mathbf{z}}_j}(\mathsf{lb}_i, \tau, 1-\bar{\mathbf{z}}_\tau) = 0$ for all $j \neq \tau$, and $C_{\bar{L},\tau,\bar{\mathbf{z}}_j}(\mathsf{lb}_i, \tau, 1-\bar{\mathbf{z}}_\tau) = 0$, because $\bar{\mathbf{z}}_\tau \neq 1-\bar{\mathbf{z}}_\tau$.

This establishes the admissibility of $\mathcal{B}$. □

Claim 5.8.3. Hybrid_2 *and* Hybrid_3 *are statistically indistinguishable.*

Proof. Firstly, we observe that $\Pr[\mathbf{z}^* = \bar{\mathbf{z}}] = 1/2^\lambda$. Hence, with probability $1 - 1/2^\lambda$, the challenger does not abort in Hybrid_3. Next we show that if the game is not aborted, the two hybrids are statistically indistinguishable in the view of the adversary. In this case, the only difference between the two hybrids is the following: for each key query (lb, x) with $\mathsf{lb} \in L^*$, the value of $K_{i,1-\bar{z}_i}$ (i.e. K_{i,z^*_i}), where i is the first index such that $z^*_i \neq \bar{z}_i$, are computed differently. In Hybrid_2, K_{i,z^*_i} is sampled uniformly, while in Hybrid_3, it is computed as $K_{i,z^*_i} = \bigoplus_{j\in[\lambda]\setminus\{i\}}(K_{j,z^*_j} \oplus R_{j,z^*_j}) \oplus R_{i,z^*_i} \oplus \alpha^*$. However, note that if $f_0(x) = 1$, then K_{i,z^*_i} is not used in anywhere because of the change we introduced in Hybrid_2 and hence does not affect the adversary's view. On the other hand, if $f_0(x) = 0$, R_{i,z^*_i} is not used anywhere, and hence K_{i,z^*_i} is uniformly random in the adversary's view because of the randomness of R_{i,z^*_i}. □

Claim 5.8.4. Hybrid_3 *and* Hybrid_4 *are identical in the view of the adversary.*

Proof. Observe that the two hybrids differ only in the computation (simulation) of SKFE ciphertexts and secret keys. In particular,

- $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_0,\mathbf{z}^*,\alpha^*}|})$ is changed to $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_1,\mathbf{z}^*,\alpha^*}|})$ in the computation of $\mathsf{RMFE.ct}^*$. This is just a conceptual change and both the computations are exactly the same, since $|P_{f_0,\mathbf{z}^*,\alpha^*}| = |P_{f_1,\mathbf{z}^*,\alpha^*}|$.[23]

[23] we can use padding to make the sizes equal.

- For each key query (lb, x), $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*}(x, K, R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$ is changed to $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_1,\mathbf{z}^*,\alpha^*}(x, K, R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$. The change is again only conceptual and does not affect the actual computation as we argue below:
  - For $f_0(x) = f_1(x)$: there is no change.
  - For $f_0(x) \neq f_1(x)$: let $f_0(x) = 1$ and $f_1(x) = 0$. Then, $\mathsf{lb} \in L^*$ and thus

$$\begin{aligned}
P_{f_0,\mathbf{z}^*,\alpha^*}(x, K, R) &= \bigoplus_{j\in[\lambda]} R_{j,z_j^*} \\
P_{f_1,\mathbf{z}^*,\alpha^*}(x, K, R) &= \bigoplus_{j\in[\lambda]\setminus\{i\}} K_{j,z_j^*} \oplus \alpha^* \oplus K_{i,z_i^*} \\
&= \bigoplus_{j\in[\lambda]\setminus\{i\}} K_{j,z_j^*} \oplus \alpha^* \oplus (\bigoplus_{j\in[\lambda]\setminus\{i\}} (K_{j,z_j^*} \oplus R_{j,z_j^*}) \oplus R_{i,z_i^*}) \oplus \alpha^* \\
&= \bigoplus_{j\in[\lambda]} R_{j,z_j^*}.
\end{aligned}$$

  - The same argument works for $f_0(x) = 0$ and $f_1(x) = 1$.

$\square$

Indistinguishability between the rest of the hybrids can be argued in the same way as their counterparts in the previous set of hybrids. In particular, proofs for indistinguishability between Hybrid_4 and Hybrid_5 is same as the proof of claim 5.8.3, Hybrid_5 and Hybrid_6 is same as the proof for claim 5.8.2 and Hybrid_6 and Hybrid_7 is same as the proof for claim 5.8.1. $\square$

6 Secret Key RPE from Evasive and Tensor LWE

In this section we construct a secret-key RPE scheme from evasive and tensor LWE, followed by the efficiency and security analysis of our scheme.

6.1 Construction

We give a construction of the secret-key RPE scheme $\mathsf{RPE} = (\mathsf{RPE.Setup}, \mathsf{RPE.KeyGen}, \mathsf{RPE.Broadcast}, \mathsf{RPE.Enc}, \mathsf{RPE.Dec})$ for an attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_\lambda$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$. We assume that the size of $|\mathcal{F}_\lambda|$ is bounded by some polynomial in λ, which will suffice for our purpose. Our scheme uses the following building blocks:

1. An RMFE scheme $\mathsf{RMFE} = (\mathsf{RMFE.Setup}, \mathsf{RMFE.KeyGen}, \mathsf{RMFE.PK\text{-}Enc}, \mathsf{RMFE.SK\text{-}Enc}, \mathsf{RMFE.Dec})$ with attribute space $\mathcal{X}$, label space $\mathcal{L}$ and function family $\mathcal{F}$. We instantiate it by our construction in Sec. 5.2 based on LWE.

2. A CP-ABE scheme $\mathsf{cpABE} = (\mathsf{cpABE.Setup}, \mathsf{cpABE.Enc}, \mathsf{cpABE.KeyGen}, \mathsf{cpABE.Dec})$ with message space $\mathcal{M}$ satisfying Sel-IND security (Def. 2.14) that supports the circuit class $\mathcal{C}_{\ell(\lambda),d(\lambda)}$ and the efficiency property listed in Theorem 2.18. We set $\ell(\lambda) = |\mathsf{RMFE.sk}_{\mathsf{lb},x}|$ and $d(\lambda)$ to be the upper bound on the depth of the circuit $C_{L,\mathsf{RMFE.ct}}$ in Eq. (6.1). Looking ahead, we show that the depth is bounded by some polynomial $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$. We choose d so that it is larger than the value. We can instantiate the scheme by the one proposed by [Wee22] based on tensor and evasive LWE.

We describe our construction below.

$\mathsf{RPE.Setup}(1^\lambda) \to (\mathsf{RPE.mpk}, \mathsf{RPE.msk})$. The setup algorithm takes as input the security parameter λ and does the following:

- Generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk}) \leftarrow \mathsf{RMFE.Setup}(1^\lambda)$ and $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$.
- Outputs $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and $\mathsf{RPE.msk} = (\mathsf{RMFE.msk}, \mathsf{cpABE.msk})$.

$\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \mathsf{lb}, x) \to \mathsf{RPE.sk}_{\mathsf{lb},x}$. The key generation algorithm takes as input the master secret key $\mathsf{RPE.msk}$, a label $\mathsf{lb} \in \mathcal{L}$ and an attribute $x \in \mathcal{X}$ and does the following:

- Parse $\mathsf{RPE.msk} = (\mathsf{RMFE.msk}, \mathsf{cpABE.msk})$.
- Compute $\mathsf{RMFE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{RMFE.KeyGen}(\mathsf{RMFE.msk}, \mathsf{lb}, x)$.
- Set $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$.
- Compute $\mathsf{cpABE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathsf{att})$.
- Output $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$.

$\mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L) \to \mathsf{RPE.ct}$. The broadcast algorithm takes as input the master public key $\mathsf{RPE.mpk}$, a message m, and a revocation list $L \subseteq \mathcal{L}$ and does the following:

- Parse $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$.
- Compute $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$.
- Construct circuit $C_{L,\mathsf{RMFE.ct}}$, with L and RMFE.ct hardwired, defined as follows: On input (lb, x) and $\mathsf{RMFE.sk}_{\mathsf{lb},x}$,

$$\begin{aligned} & C_{L,\mathsf{RMFE.ct}}((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}) \\ & \quad =(\mathsf{lb} \notin L) \wedge (\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) = 1). \end{aligned} \tag{6.1}$$

- Compute $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}}, m)$.
- Output $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$.

$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L) \to \mathsf{RPE.ct}$. The encryption algorithm takes as input the master secret key, a function f, a message m, and a revocation list $L \subseteq \mathcal{L}$ and does the following:

- Parse $\mathsf{RPE.msk} = (\mathsf{RMFE.msk}, \mathsf{cpABE.msk})$.
- Compute $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$.
- Construct circuit $C_{L,\mathsf{RMFE.ct}}$ from L and RMFE.ct as defined in Eq. (6.1).
- Compute $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}}, m)$.
- Output $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$.

$\mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},x}, \mathsf{RPE.ct}, L) \to m$. The decryption algorithm takes as input the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$, a ciphertext RPE.ct, and a revocation list L and does the following:

- Parse $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}), \mathsf{cpABE.sk}_{\mathsf{lb},x})$ and $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$.
- Construct circuit $C_{L,\mathsf{RMFE.ct}}$ from L and RMFE.ct as defined in Eq. (6.1).
- Compute and output $\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathsf{lb},x}, \mathsf{att}, \mathsf{cpABE.ct}, C_{L,\mathsf{RMFE.ct}})$.

Correctness. First, we show that $C_{L,\mathsf{RMFE.ct}} \in \mathcal{C}_{\ell,d}$. In particular, it suffices to bound the depth of the circuit d by some fixed polynomial $\mathrm{poly}(\lambda)$. We first observe that checking whether $\mathsf{lb} \in L$ or not can be done with depth $|\log(|\mathsf{lb}| \cdot |L|) = \log(\mathrm{poly}(\lambda)) \leq \lambda$. We also have that the depth of $\mathsf{RMFE.Dec}$ is bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ as we saw in Sec. 5.2. Therefore, the total depth is bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

Theorem 6.1. *Suppose* RMFE *and* cpABE *are correct, then the above construction of secret-key* RPE *satisfies correctness (Def. 3.2).*

Proof. For any (lb, x), function $f \in \mathcal{F}_\lambda$ and a revocation list $L \subseteq \mathcal{L}_\lambda$ such that $f(x) = 1, \forall x \in \mathcal{X}_\lambda$ and $\mathsf{lb} \notin L$, consider the following two cases:

1. **Broadcast Correctness:** For any ciphertext $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L)$, we have $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$, where $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$. So, by the public encryption correctness of RMFE scheme, with all but negligible probability
$$\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) = 1.$$
Since $\mathsf{lb} \notin L$, we have $C_{L,\mathsf{RMFE.ct}}(\mathsf{att}) = 1$, where $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$. Hence by the correctness of cpABE scheme we have that with all but negligible probability
$$\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathsf{lb},x}, \mathsf{att}, \mathsf{cpABE.ct}, C_{L,\mathsf{RMFE.ct}}) = m.$$

2. **Encryption Correctness:** For any ciphertext $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L)$, we have $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$, where $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$. If $(\mathsf{lb} \notin L) \wedge f(x) = 1$, then by the correctness of RMFE.SK-Enc algorithm, we have with all but negligible probability
$$\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) = 1.$$
Furthermore, $\mathsf{lb} \notin L$ implies $C_{L,\mathsf{RMFE.ct}}((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}) = 1$. Hence by the correctness of cpABE scheme, we have that with all but negligible probability
$$\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathsf{lb},x}, \mathsf{att}, \mathsf{cpABE.ct}, C_{L,\mathsf{RMFE.ct}}) = m.$$

Hence the above construction of secret-key RPE satisfies correctness. □

Efficiency. Here we argue that our construction achieves optimal parameters. Namely, we show that all the parameters are independent from $|L|$.

1. Public key size $|\mathsf{RPE.mpk}|$: We have $|\mathsf{RPE.mpk}| = |\mathsf{cpABE.mpk}| + |\mathsf{RMFE.mpk}|$. The former is bounded by $\mathrm{poly}(\lambda, \ell, d) = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ using Theorem 2.18, where we additionally used $\ell = |\mathsf{RMFE.sk}_{\mathsf{lb},x}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ and $d = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

2. Secret key size $|\mathsf{RPE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RPE.sk}_{\mathsf{lb},x}| = |\mathsf{att}| + |\mathsf{cpABE.sk}_{\mathsf{lb},x}|$. We have $|\mathsf{att}| = |\mathsf{lb}| + |x| + |\mathsf{RMFE.sk}_{\mathsf{lb},x}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ and $|\mathsf{cpABE.sk}_{\mathsf{lb},x}| = \mathrm{poly}(\ell, d) = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ by Theorem 2.18. Therefore, the overall size is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

3. Ciphertext size $|\mathsf{RPE.ct}|$: We have $|\mathsf{RPE.ct}| = |\mathsf{RMFE.ct}| + |\mathsf{cpABE.ct}|$. The former is bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ and the latter is bounded by $\mathrm{poly}(\lambda, d) = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ by Theorem 2.18. Therefore, the overall size is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

6.2 Security

In this section we show that our construction of secret-key RPE is secure.

Message Hiding Security

Theorem 6.2. *Assume that* cpABE *is selectively secure (Def. 2.14),* RMFE *is correct and* $|\mathcal{F}| \leq \text{poly}(\lambda)$*. Then* RPE *scheme satisfies* 1*-query selective message hiding security (Def. 3.4).*

Proof. Recall that in the message hiding security game, we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_0, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_1, L),$$

where for all the key queries (lb, x) to the $\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \cdot, \cdot)$ oracle, either $f(x) = 0$ or $\mathsf{lb} \in L$. We show that if there exists an adversary $\mathcal{A}$ who has non-negligible advantage ϵ in the selective message hiding security game, then there exists a PPT adversary $\mathcal{B}$ against the security of cpABE scheme with the same advantage ϵ. The reduction is as follows.

1. $\mathcal{B}$ first runs $\mathcal{A}$ and gets f and L[24].
2. $\mathcal{B}$ generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk}) \leftarrow \mathsf{RMFE.Setup}(1^\lambda)$ and computes $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$.
3. $\mathcal{B}$ constructs the circuit $C_{L,\mathsf{RMFE.ct}}$ as defined in the construction and sends it as the challenge function to the cpABE challenger. The cpABE challenger generates $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ and returns $\mathsf{cpABE.mpk}$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and sends it to $\mathcal{A}$.
4. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:
   - Computes $\mathsf{RMFE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{RMFE.KeyGen}(\mathsf{RMFE.msk}, \mathsf{lb}, x)$.
   - Sets $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$, sends it as a key query to cpABE challenger and gets back $\mathsf{cpABE.sk}_{\mathsf{lb},x}$.
   - It returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$ to $\mathcal{A}$.
5. **Challenge Query:** When $\mathcal{A}$ sends the challenge messages (m_0, m_1), $\mathcal{B}$ forwards it to the cpABE challenger. The cpABE challenger samples a bit $\hat{\beta} \leftarrow \{0, 1\}$ and returns $\mathsf{cpABE.ct}_{\hat{\beta}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}}, m_{\hat{\beta}})$ to $\mathcal{B}$. $\mathcal{B}$ sends $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct}_{\hat{\beta}})$ to $\mathcal{A}$.
6. **Encryption Query:** When $\mathcal{A}$ makes the encryption query $(\bar{f}, \bar{m}, \bar{L})$, $\mathcal{B}$ does the following:
   - Computes $\mathsf{RMFE.\bar{ct}} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, \bar{f}, \bar{L})$ and constructs the circuit $C_{\bar{L},\mathsf{RMFE.\bar{ct}}}$ as defined in the construction.
   - Computes $\mathsf{cpABE.\bar{ct}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{\bar{L},\mathsf{RMFE.\bar{ct}}}, \bar{m})$.
   - Returns $\mathsf{RPE.\bar{ct}} = (\mathsf{RMFE.\bar{ct}}, \mathsf{cpABE.\bar{ct}})$ to $\mathcal{A}$.
7. In the end, $\mathcal{A}$ outputs a bit β'.

[24] To keep the proofs simple, we let f to be given selectively. This is sufficient to achieve security as in Def 3.4, as mentioned in Remark 3.9.

Observe that if the cpABE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the real world where m_0 was encrypted, else it simulated the real world where m_1 was encrypted with $\mathcal{A}$.
Hence, advantage of $\mathcal{B}$ is $|\Pr[\beta' = 1|\hat{\beta} = 0] - \Pr[\beta' = 1|\hat{\beta} = 1]| = |\Pr[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_0, L)] - \Pr[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_1, L)]| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. We observe that for the challenge circuit $C_{L,\mathsf{RMFE.ct}}$ and for all key queries (lb, x), queried by $\mathcal{B}$, we have $C_{L,\mathsf{RMFE.ct}}((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}) = 0$ as either (i) $\mathsf{lb} \in L$, or (ii) $\mathsf{lb} \in L$ and $f(x) = 0$, which implies $\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}) = 0$ (due to RMFE correctness), by the admissibility condition on $\mathcal{A}$. So, if $\mathcal{A}$ is admissible then so is $\mathcal{B}$. □

Function Hiding Security

Theorem 6.3. *Assume* RMFE *satisfies* 1*-query selective function hiding security (Def. 5.3), then the* RPE *scheme satisfies* 1*-query selective function hiding security (Def. 3.6).*

Proof. Recall that in the function hiding security game, we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_0, m, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_1, m, L),$$

where for all the key queries (lb, x) to the $\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \cdot, \cdot)$ oracle, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.
We show that if there exists an adversary $\mathcal{A}$ who has non-negligible advantage ϵ in the 1-query selective function hiding security game, then there exists a PPT adversary $\mathcal{B}$ against the 1-query selective function-hiding security of RMFE scheme with the same advantage ϵ. The reduction is as follows.

1. $\mathcal{B}$ runs $\mathcal{A}$ and gets the revocation list L.
2. $\mathcal{B}$ sends L as the challenge revocation list to the RMFE challenger. The challenger generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk})$ and returns $\mathsf{RMFE.mpk}$ to $\mathcal{B}$.
3. $\mathcal{B}$ generates $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and sends it to $\mathcal{A}$.
4. **Key Queries:** On each key query (lb, x), $\mathcal{B}$ does the following:
   - It sends a key query (lb, x) to the RMFE challenger and gets back $\mathsf{RMFE.sk}_{\mathsf{lb},x}$.
   - Sets $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$ and computes $\mathsf{cpABE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathsf{att})$.
   - It returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$ to $\mathcal{A}$.
5. **Challenge Query:** When $\mathcal{A}$ sends the challenge functions (f_0, f_1) and message m, $\mathcal{B}$ does the following:
   - Sends (f_0, f_1) as the challenge functions to the RMFE challenger. The challenger samples $\hat{\beta} \leftarrow \{0, 1\}$, computes $\mathsf{RMFE.ct}_{\hat{\beta}} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f_{\hat{\beta}}, L)$ and returns $\mathsf{RMFE.ct}_{\hat{\beta}}$ to $\mathcal{B}$.
   - Constructs the circuit $C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}$ as defined in the construction and computes $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}, m)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}_{\hat{\beta}}, \mathsf{cpABE.ct})$ to $\mathcal{A}$.

6. **Encryption Query:** When $\mathcal{A}$ makes an encryption query $(\bar{f}, \bar{m}, \bar{L})$, $\mathcal{B}$ does the following:
   - Sends a SK-Enc query $(\bar{f}, \bar{L})$ to the RMFE challenger and gets back $\mathsf{RMFE}.\bar{\mathsf{ct}}$.
   - Constructs the circuit $C_{\bar{L},\mathsf{RMFE}.\bar{\mathsf{ct}}}$ as defined in the construction.
   - Computes $\mathsf{cpABE}.\bar{\mathsf{ct}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{\bar{L},\mathsf{RMFE}.\bar{\mathsf{ct}}}, \bar{m})$.
   - Returns $\mathsf{RPE}.\bar{\mathsf{ct}} = (\mathsf{RMFE}.\bar{\mathsf{ct}}, \mathsf{cpABE}.\bar{\mathsf{ct}})$ to $\mathcal{A}$.
7. In the end, the adversary outputs a bit β'.

Observe that if the RMFE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the real world where f_0 was encrypted, else it simulated the real world where f_1 was encrypted with $\mathcal{A}$.
Hence, advantage of $\mathcal{B}$ is $|\Pr[\beta' = 1|\hat{\beta} = 0] - \Pr[\beta' = 1|\hat{\beta} = 1]| = |\Pr[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_0, m, L)] - \Pr[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_1, m, L)]| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. First, we note that since $\mathcal{A}$ is allowed to make only one query, $(\bar{f}, \bar{m}, \bar{L})$ to the $\mathsf{RPE.Enc}(\mathsf{msk}, \cdot, \cdot, \cdot)$ oracle, $\mathcal{B}$ also makes only one query, $(\bar{f}, \bar{L})$, to the $\mathsf{RMFE.SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle. Next, we observe that since $\mathcal{A}$ is restricted to make key queries (lb, x) such that either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$, thus $\mathcal{B}$ also issues key queries (lb, x) to RMFE challenger such that either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$. Hence, if $\mathcal{A}$ is admissible, then so is $\mathcal{B}$. □

Broadcast Security

Theorem 6.4. *Assume* RMFE *satisfies* 1*-query mode hiding security (Def. 5.2), then the* RPE *scheme satisfies* 1*-query selective broadcast security (Def. 3.7).*

Proof. Recall that in the broadcast security game, we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L) \approx_c \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L),$$

where $f(x) = 1, \forall x \in \mathcal{X}$.
We show that if there exists an adversary $\mathcal{A}$ who has non-negligible advantage ϵ in the broadcast security game, then there exists a PPT adversary $\mathcal{B}$ against the mode-hiding security of RMFE scheme with the same advantage ϵ. The reduction is as follows.

1. $\mathcal{B}$ runs $\mathcal{A}$ and gets the revocation list L.
2. $\mathcal{B}$ sends L as the challenge revocation list to the RMFE challenger. The challenger generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk})$ and returns $\mathsf{RMFE.mpk}$ to $\mathcal{B}$.
3. $\mathcal{B}$ generates $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and sends it to $\mathcal{A}$.
4. **Key Queries:** On each key query (lb, x), $\mathcal{B}$ does the following:
   - It sends a key query (lb, x) to the RMFE challenger and gets back $\mathsf{RMFE.sk}_{\mathsf{lb},x}$.
   - Sets $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$ and computes $\mathsf{cpABE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathsf{att})$.
   - It returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$ to $\mathcal{A}$.
5. **Challenge Query:** When $\mathcal{A}$ sends the challenge function f and message m, $\mathcal{B}$ does the following:

- Sends f as the challenge function to the RMFE challenger. The challenger samples $\hat{\beta} \leftarrow \{0,1\}$, and returns $\mathsf{RMFE.ct}_{\hat{\beta}}$ to $\mathcal{B}$, where $\mathsf{RMFE.ct}_0 \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$ and $\mathsf{RMFE.ct}_1 \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$.
- Constructs the circuit $C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}$ as defined in the construction and computes $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}, m)$.
- Returns $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}_{\hat{\beta}}, \mathsf{cpABE.ct})$ to $\mathcal{A}$.

6. **Encryption Query:** When $\mathcal{A}$ makes an encryption query $(\bar{f}, \bar{m}, \bar{L})$, $\mathcal{B}$ does the following:
   - Sends a SK-Enc query $(\bar{f}, \bar{L})$ to the RMFE challenger and gets back $\mathsf{RMFE}.\bar{\mathsf{ct}}$.
   - Constructs the circuit $C_{\bar{L},\mathsf{RMFE}.\bar{\mathsf{ct}}}$ as defined in the construction.
   - Computes $\mathsf{cpABE}.\bar{\mathsf{ct}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{\bar{L},\mathsf{RMFE}.\bar{\mathsf{ct}}}, \bar{m})$.
   - Returns $\mathsf{RPE}.\bar{\mathsf{ct}} = (\mathsf{RMFE}.\bar{\mathsf{ct}}, \mathsf{cpABE}.\bar{\mathsf{ct}})$ to $\mathcal{A}$.
7. In the end, the adversary outputs a bit β'.

Observe that if the RMFE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the real world where the ciphertext was computed using Enc algorithm else it simulated the real world where the ciphertext was computed using Broadcast algorithm, with $\mathcal{A}$.
Hence, the advantage of $\mathcal{B}$ is $|\Pr[\beta' = 1|\hat{\beta} = 0] - \Pr[\beta' = 1|\hat{\beta} = 1]| = |\Pr[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L)] - \Pr[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Broadcast}(\mathsf{RPE.msk}, m, L)]| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. First, we note that since $\mathcal{A}$ is allowed to make only one query, $(\bar{f}, \bar{m}, \bar{L})$ to the $\mathsf{RPE.Enc}(\mathsf{msk}, \cdot, \cdot, \cdot)$ oracle, $\mathcal{B}$ also makes only one query, $(\bar{f}, \bar{L})$, to the $\mathsf{RMFE.SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle. Next, we observe that since $\mathcal{A}$ is restricted to output f, such that $f(x) = 1$ for all $x \in \mathcal{X}$, this implies $\mathcal{B}$ also issues such f as the challenge query to the RMFE challenger in the above mode hiding security game. Thus, if $\mathcal{A}$ is admissible, then so is $\mathcal{B}$. $\square$

7 Embedded Identity Trace and Revoke

In this section, we define different variants of an embedded identity trace and revoke system (EITR). Our definitions extend the different notions that Goyal et al. [GKW19] introduced, in the context of embedded identity traitor tracing, to incorporate the revocation list. Concretely, we define three variants of an EITR scheme: 1) Indexed EITR, 2) Bounded EITR, and 3) Unbounded EITR. Our goal is Unbounded EITR and other variants. Other variants are introduced as intermediate goals. We show a construction of EITR from RPE in Sec. 8, bounded EITR from indexed EITR in Sec. 9, and unbounded EITR from bounded EITR in Sec. 10. These implications hold for both secret key and public key settings.

An EITR scheme consists of five polynomial time algorithms–Setup, KeyGen, Enc, Dec and Trace. The syntax of the EITR variants mentioned above differs only in the inputs to the Setup, KeyGen and Trace algorithms. Here we give a unified definition and later specify the distinctness of the three variants.

Consider a general identity space $\mathcal{GID}$, a label space $\mathcal{L}$ and a message space $\mathcal{M}$. An embedded identity trace and revoke scheme $\mathsf{EITR} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ has the following syntax:

$\mathsf{Setup}(1^\lambda, \mathsf{params}_1) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and parameters params_1. It outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{gid}) \to \mathsf{sk}_{\mathsf{lb},\mathsf{gid}}$. The key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}$, and a general identity $\mathsf{gid} \in \mathcal{GID}$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. The encryption algorithm takes as input the master public key mpk, a message $m \in \mathcal{M}$, a revocation list $L \subseteq \mathcal{L}$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}, \mathsf{ct}, L) \to y$. The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}$, a ciphertext ct, and a revocation list L and outputs $y \in \mathcal{M} \cup \{\perp\}$.

$\mathsf{Trace}^D(\mathsf{tk}, \mathsf{params}_2, m_0, m_1, L) \to T$. The tracing algorithm takes as input a tracing key tk, parameter params_2, two messages m_0, m_1, a revocation list L and has an oracle access to a decoder D. It outputs a set of traitors T.

The above syntax captures both public key and secret key trace EITR schemes. For any *public tracing* EITR scheme, we have $\mathsf{tk} = \mathsf{mpk}$ and in the *secret tracing* EITR scheme, $\mathsf{tk} = \mathsf{msk}$. We now describe the properties satisfied by an EITR scheme.

Definition 7.1 (Correctness)**.** An EITR scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, any label $\mathsf{lb} \in \mathcal{L}$, and any revocation list $L \subseteq \mathcal{L}$ such that $\mathsf{lb} \notin L$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}, \mathsf{ct}, L) = m : \begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{params}_1);\\ \mathsf{sk}_{\mathsf{lb},\mathsf{gid}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{gid});\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m, L)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 7.2 (IND-CPA Security)**.** An EITR scheme is said to be IND-CPA secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}_\beta) = \beta : \begin{array}{l}\mathsf{params}_1 \leftarrow \mathcal{A}(1^\lambda);\\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{params}_1);\\ (m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk});\\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b, L)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where the adversary has the access to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle which has msk hardwired and $\mathcal{A}$ is admissible only if for all the key generation queries $(\mathsf{lb}, \mathsf{gid})$ to the KeyGen oracle, $\mathsf{lb} \in L$.

Definition 7.3 (Selective IND-CPA Security)**.** The selective IND-CPA security of an EITR scheme is defined in the same way as Def. 7.2, except that the adversary outputs the revocation list L along with params_1 before the Setup algorithm is run.

In the following, we specify the space $\mathcal{GID}$, inputs $\mathsf{params}_1, \mathsf{gid}$ and params_2 to the Setup, KeyGen and Trace algorithm, respectively, and then define the secure tracing guarantee of each of the EITR notions separately. We let $\mathcal{ID} = \{0,1\}^\kappa$ denote the identity space.

Remark 7.4. We assume that there exists an efficiently computable mapping $\mathsf{map} : \mathcal{GID} \to \mathcal{L}$, that *uniquely* maps an $\mathsf{gid} \in \mathcal{GID}$ to a label $\mathsf{lb} \in \mathcal{L}$. This can be easily ensured, for e.g. by making label lb a part of id. We further note that in real world applications one may also want to ensure that any label lb is associated with at most one id. This can be achieved by using a collision resistant hash function.

Experiment $\mathsf{Expt\text{-}Ind\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$

- $1^\kappa, 1^{n_{\mathsf{ind}}} \leftarrow \mathcal{A}(1^\lambda)$
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa, n_{\mathsf{ind}})$
- $(D, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk})$
- $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{1/\epsilon(\lambda)}, m_0, m_1, L)$

Here $\mathcal{A}$ has the oracle access to $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot, \cdot)$, which has msk hardwired and on query $(\mathsf{lb}, \mathsf{id}, i)$, it outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id},i}$. The adversary is admissible only if it makes at most one key query for each index.

Figure 3: Expt-Ind-TR

7.1 Indexed Trace and Revoke with Embedded Identity

In an indexed EITR scheme, the key is generated w.r.t. an identity and an index. We have $\mathcal{GID} = \mathcal{ID} \times [n_{\mathsf{ind}}]$, where $[n_{\mathsf{ind}}]$ is the index space for $n_{\mathsf{ind}} \in \mathbb{N}$, $\mathsf{params}_1 = (1^\kappa, n_{\mathsf{ind}})$, $\mathsf{gid} = (\mathsf{id}, i) \in \mathcal{ID} \times [n_{\mathsf{ind}}]$, and $\mathsf{params}_2 = y$ for some $y > 1$.
We now define the secure tracing requirement.

Definition 7.5 (Secure Tracing)**.** Let $\mathsf{Ind\text{-}TR} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be an indexed EITR scheme. For any non-negligible function $\epsilon(\cdot)$ and stateful PPT adversary $\mathcal{A}$, define an experiment $\mathsf{Expt\text{-}Ind\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$ as in Figure 3.
Let S be the set of key queries $(\mathsf{lb}, \mathsf{id}, i)$ queried by $\mathcal{A}$ and $S_{\mathcal{ID}} = \{\mathsf{id} : \exists \mathsf{lb} \in \mathcal{L}, i \in [n_{\mathsf{ind}}] \text{ s.t } (\mathsf{lb}, \mathsf{id}, i) \in S\}$.
Consider the following probabilistic events and their corresponding probabilities:

- $\mathsf{Good\text{-}Decoder} : \Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b, L)] \geq 1/2 + \epsilon(\lambda)$
  $\mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder}]$.
- $\mathsf{Cor\text{-}Tr} : |T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$, where $T_{\mathsf{lb}} = \{\mathsf{lb} \in \mathcal{L} : \exists i \in [n_{\mathsf{ind}}], \exists \mathsf{id} \in T, (\mathsf{lb}, \mathsf{id}, i) \in S\}$.
  $\mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- $\mathsf{Fal\text{-}Tr} : (T \nsubseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$
  $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

The $\mathsf{Ind\text{-}TR}$ scheme is said to satisfy secure tracing property if for every PPT adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda), \quad \mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \geq \mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda).$$

Definition 7.6 (Selective Secure Tracing)**.** The selective secure tracing of an indexed EITR scheme is defined in the same way as Def. 7.5, except that the adversary outputs the challenge revocation list L along with $(1^\kappa, 1^{n_{\mathsf{ind}}})$ in the beginning of the $\mathsf{Expt\text{-}Ind\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$.

Remark 7.7. In the definition above, Setup takes n_{ind} as an input in the binary form rather than in the unary form. This indicates that Setup runs in polynomial time, even if $n_{\mathsf{ind}} = 2^{\mathrm{poly}(\lambda)}$. On

Experiment $\mathsf{Expt\text{-}BD\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$

- $1^{\kappa}, 1^{n_{\mathsf{bd}}} \leftarrow \mathcal{A}(1^{\lambda})$
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\kappa}, n_{\mathsf{bd}})$
- $(D, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk})$
- $T \leftarrow \mathsf{Trace}^{D}(\mathsf{tk}, 1^{1/\epsilon}, m_0, m_1, L)$

Here $\mathcal{A}$ has the oracle access to $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$, which has msk hardwired and on query $(\mathsf{lb}, \mathsf{id})$, it outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id}}$.

Figure 4: Expt-BD-TR

the other hand, the adversary outputs n_{ind} in the unary form in the security game (Fig. 3). This indicates that we consider the security game only for the case where $n_{\mathsf{ind}} = \mathrm{poly}(\lambda)$. Looking ahead, the former property is necessary when we convert bounded EITR into unbounded EITR in Sec. 10.

7.2 Bounded Trace and Revoke with Embedded Identity

In a bounded EITR scheme, we have $\mathcal{GID} = \mathcal{ID}$, $\mathsf{params}_1 = (1^{\kappa}, 1^{n_{\mathsf{bd}}})$, where $n_{\mathsf{bd}} \in \mathbb{N}$ is the bound on the number of key queries that an adversary can make in the *correct trace experiment* game, $\mathsf{gid} = \mathsf{id}$ for $\mathsf{id} \in \mathcal{ID}$, and $\mathsf{params}_2 = y$ for some $y > 1$.
We now define the secure tracing requirement.

Definition 7.8 (Secure Tracing)**.** Let BD-TR = (Setup, KeyGen, Enc, Dec, Trace) be a bounded EITR scheme. For any non-negligible function $\epsilon(\cdot)$ and stateful PPT adversary $\mathcal{A}$, define an experiment $\mathsf{Expt\text{-}BD\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$ as in Figure 4.
Let S be the set of key queries made by $\mathcal{A}$ and $S_{\mathcal{ID}} = \{\mathsf{id} : \exists \mathsf{lb} \in \mathcal{L} \text{ s.t } (\mathsf{lb}, \mathsf{id}) \in S\}$. Consider the following probabilistic events and their corresponding probabilities :

- Good-Decoder : $\Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b, L)] \geq 1/2 + \epsilon(\lambda)$
  $\mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder} \wedge |S_{\mathcal{ID}}| \leq n_{\mathsf{ind}}]$.
- Cor-Tr : $|T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$, where $T_{\mathsf{lb}} = \{\mathsf{lb} \in \mathcal{L} : \exists \mathsf{id} \in T, (\mathsf{lb}, \mathsf{id}) \in S\}$.
  $\mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- Fal-Tr : $(T \not\subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$
  $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

The scheme is said to satisfy secure tracing if for every PPT adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda), \qquad \mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \geq \mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda).$$

Definition 7.9 (Selective Secure Tracing)**.** The selective secure tracing of a bounded EITR scheme is defined in the same way as Def. 7.8, except that the adversary outputs the challenge revocation list L along with $(1^{\kappa}, 1^{n_{\mathsf{bd}}})$ in the beginning of the $\mathsf{Expt\text{-}BD\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$.

Experiment $\mathsf{Expt\text{-}TR}_{\mathcal{A},\epsilon,\mathsf{p}}(\lambda)$

- $1^\kappa \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$.
- $(D, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk})$.
- $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{1/\epsilon(\lambda)}, p(\lambda), m_0, m_1, L)$.

Here $\mathcal{A}$ has the oracle access to $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$, which has msk hardwired and on query $(\mathsf{lb}, \mathsf{id})$, it outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id}}$.

Figure 5: Expt-TR

Remark 7.10. We point out that the bound on the number of key queries by the adversary is only required for the correct trace guarantee. The false trace guarantee will hold even if the adversary exceeds n_{bd} key queries– this is essential for the transformation from bounded-EITR scheme to unbounded-EITR scheme.

7.3 Unbounded Trace and Revoke with Embedded Identity

In an unbounded EITR scheme, we have $\mathcal{GID} = \mathcal{ID}$, $\mathsf{params}_1 = 1^\kappa$, $\mathsf{gid} = \mathsf{id}$ for $\mathsf{id} \in \mathcal{ID}$, and $\mathsf{params}_2 = (y, \mathrm{Q}_{\mathsf{bd}})$ where $y > 1, \mathrm{Q}_{\mathsf{bd}} \in \mathbb{N}$.
We now define the secure tracing requirement.

Definition 7.11 (Secure Tracing)**.** Let $\mathsf{TR} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be an unbounded EITR scheme. For any non-negligible function $\epsilon(\cdot)$, polynomial $p(\cdot)$ and stateful PPT adversary $\mathcal{A}$, define the experiment $\mathsf{Expt\text{-}TR}_{\mathcal{A},\epsilon,\mathsf{p}}(\lambda)$ as in Figure 5.
Let S be the set of key queries made by $\mathcal{A}$ and $S_{\mathcal{ID}} = \{\mathsf{id} : \exists \mathsf{lb} \in \mathcal{L} \text{ s.t } (\mathsf{lb}, \mathsf{id}) \in S\}$. Consider the following probabilistic events and their corresponding probabilities :

- $\mathsf{Good\text{-}Decoder}$: $\Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b, L) \geq 1/2 + \epsilon(\lambda)]$
  $\mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon,p}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder} \wedge |S_{\mathcal{ID}}| \leq p(\lambda)]$.
- $\mathsf{Cor\text{-}Tr}$: $|T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$, where $T_{\mathsf{lb}} = \{\mathsf{lb} \in \mathcal{L} : \exists \mathsf{id} \in T, (\mathsf{lb}, \mathsf{id}) \in S\}$.
  $\mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- $\mathsf{Fal\text{-}Tr}$: $(T \nsubseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$
  $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

The scheme is said to satisfy secure traitor tracing property if for every PPT adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$,there exists negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) \leq \mathsf{negl}_1(\lambda), \quad \mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) \geq \mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon,p}(\lambda) - \mathsf{negl}_2(\lambda).$$

Definition 7.12 (Selective Secure Tracing)**.** The selective secure tracing of an unbounded EITR scheme is defined in the same way as Def. 7.11, except that the adversary outputs the challenge revocation list L along with 1^κ in the beginning of the $\mathsf{Expt\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$.

Inputs: an identity id and an index i.
Hardwired Values : Indices $j \in [n_{\mathsf{ind}}]$, $\ell \in [\kappa]$, a bit $b \in \{0,1\}$.
Output : $0/1$.

$$f[j,\ell,b](\mathsf{id},i) = \begin{cases} 1 & \text{if } (i > j) \vee (i = j \wedge \ell = \perp) \vee (i = j \wedge \mathsf{id}_\ell = 1 - b) \\ 0 & \text{otherwise} \end{cases}$$

Figure 6: Comparison Function $f[j,\ell,b]$

Remark 7.13. [Remark 10.4, [GKW19]]. Note that here the trace algorithm takes an additional parameter Q_{bd}. In the correct trace definition, we require that as long as the tracing algorithm uses a bound greater than the number of keys queried, the tracing algorithm must identify at least one traitor. However, the false trace guarantee should hold for all polynomially bounded Q_{bd} values. In particular, even if the number of keys queried is more than the bound used in tracing, the trace algorithm must not output an identity that was not queried. We can show that this definition implies the 'standard' tracing definition where the trace algorithm does not take this bound as input. One simply needs to run this bounded-version of trace with increasing powers of two until the trace algorithm outputs at least one traitor.

8 Indexed Trace and Revoke with Embedded Identity

In this section we construct an indexed (secret/public tracing)-EITR scheme from a (secret/public key)-RPE scheme. We present the construction and proofs for the secret trace setting primarily and also outline the differences in the public trace setting simultaneously.

8.1 Construction

Consider the identity space $\mathcal{ID} = \{0,1\}^\kappa$ and index bound n_{ind}. For our purpose, it suffices to assume $n_{\mathsf{ind}} \leq 2^{2\lambda}$. Let RPE = (RPE.Setup, RPE.KeyGen, RPE.Broadcast, RPE.Enc, RPE.Dec) be a (secret/public key) RPE scheme[25] with attribute space $\mathcal{ID} \times [n_{\mathsf{ind}}]$, and supports the function class $\mathcal{F} = \{f[j,\ell,b]\}_{\substack{j \in [n_{\mathsf{ind}}], \ell \in [\kappa], \\ b \in \{0,1\}}}$, where $f[j,\ell,b] : \mathcal{ID} \times [n_{\mathsf{ind}}] \to \{0,1\}$ is as defined in figure 8.1.
We construct an indexed (secret/public tracing)-EITR scheme with identity space $\mathcal{ID} = \{0,1\}^\kappa$ and index bound n_{ind} as follows :

$\mathsf{Setup}(1^\lambda, 1^\kappa, n_{\mathsf{ind}}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following:

- Samples $(\mathsf{RPE.mpk}, \mathsf{RPE.msk}) \leftarrow \mathsf{RPE.Setup}(1^\lambda)$.
- Outputs $\mathsf{mpk} = \mathsf{RPE.mpk}$ and $\mathsf{msk} = \mathsf{RPE.msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, (\mathsf{lb}, \mathsf{id}, i)) \to \mathsf{sk}_{\mathsf{lb},\mathsf{id},i}$. The key generation algorithm does the following:

- Parse $\mathsf{msk} = \mathsf{RPE.msk}$.
- Sets $x = (\mathsf{id}, i)$ and runs $\mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i} \leftarrow \mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \mathsf{lb}, x)$.
- Outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id},i} = \mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i}$.

[25] Public key RPE does not have the Broadcast algorithm.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. The encryption algorithm does the following:

- Parse $\mathsf{mpk} = \mathsf{RPE.mpk}$.
- Compute $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L)$. (In public trace setting, the algorithm computes $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1, \perp, 0], m, L)$).
- Outputs $\mathsf{ct} = \mathsf{RPE.ct}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{id},i}, \mathsf{ct}, L) \to m'$. The decryption algorithm does the following:

- Parse $\mathsf{sk}_{\mathsf{lb},\mathsf{id},i}$ as $\mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i}$ and ct as $\mathsf{RPE.ct}$.
- Computes and outputs $m' \leftarrow \mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i}, \mathsf{RPE.ct}, L)$.

$\mathsf{Trace}^D(\mathsf{tk}, y, m_0, m_1, L) \to T$. Parse tk as $\mathsf{msk} = \mathsf{RPE.msk}$ (in the public trace setting tk is $\mathsf{mpk} = \mathsf{RPE.mpk}$). The tracing algorithm is a two phased process. Briefly the tracing is implemented as follows:

1. First, we run the $\mathsf{Index.Trace}$ algorithm as defined in Figure 7, on the index space $[n_{\mathsf{ind}}]$. If the key associated with index $i \in [n_{\mathsf{ind}}]$ is used in constructing the decoder box, then the bit b in the output of the $\mathsf{Index.Trace}$ algorithm is 1. We maintain a set T_{index} to store all such indices on which $b = 1$.

2. Next, the $\mathsf{ID.Trace}$ algorithm, defined in Figure 8, takes the set T_{index} as input and uses the decoder D to compute the identity associated with each index.

Specifically, the algorithm runs as follows:

- Set $T_{\mathsf{index}} := \phi$. For $i = 1$ to n_{ind},
  - Compute $(b, p, q) \leftarrow \mathsf{Index.Trace}(\mathsf{tk}, y, m_0, m_1, L, i)$.
  - If $b = 1$, set $T_{\mathsf{index}} := T_{\mathsf{index}} \cup (i, p, q)$.
- Set $T = \phi$. For $(i, p, q) \in T_{\mathsf{index}}$,
  - Compute $\mathsf{id} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, y, m_0, m_1, L, (i, p, q))$.
  - Set $T := T \cup \{\mathsf{id}\}$.
- Output T.

Correctness. We prove that the above construction of indexed (secret/public tracing)-EITR scheme satisfies correctness (Def. 7.1) via the following theorem.

Theorem 8.1. *If* RPE *is a correct (secret/public key)-*RPE *scheme, then the above construction of indexed (secret/public tracing)-*EITR *scheme is correct.*

Proof. For the secret trace setting, the correctness follows directly from the broadcast correctness of the underlying (secret-key) RPE scheme.
For the public trace setting, firstly we observe that for any $(\mathsf{id}, i) \in \mathcal{ID} \times [n_{\mathsf{ind}}]$, $f[1, \perp, 0](\mathsf{id}, i) = 1$. Hence, correctness follows directly from the encryption correctness of the underlying (public-key) RPE scheme.

□

Algorithm $\mathsf{Index.Trace}(\mathsf{tk}, y, m_0, m_1, L, i)$

Inputs: Tracing key tk, a parameter y, messages m_0, m_1, revocation list L and an index i.
Output : (b, p, q), where $b \in \{0,1\}$ and $p, q \in [0,1] \cup \{\perp\}$.
Let $\epsilon = \lfloor 1/y \rfloor$. It sets $N = \lambda \cdot n_{\mathsf{ind}}/\epsilon$ and $\mathsf{temp}_1 = \mathsf{temp}_2 = 0$. For $j = 1$ to N, it computes the following:

1. It samples $b_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_{j,1} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \perp, 0], m_{b_j}, L)$ and sends $\mathsf{ct}_{j,1}$ to D.
   If D outputs b_j, set $\mathsf{temp}_1 = \mathsf{temp}_1 + 1$, else set $\mathsf{temp}_1 = \mathsf{temp}_1 - 1$.

2. It samples $c_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_{j,2} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i+1, \perp, 0], m_{c_j}, L)$ and sends $\mathsf{ct}_{j,2}$ to D.
   If D outputs c_j, set $\mathsf{temp}_2 = \mathsf{temp}_2 + 1$, else set $\mathsf{temp}_2 = \mathsf{temp}_2 - 1$.

If $\frac{\mathsf{temp}_1 - \mathsf{temp}_2}{N} > \frac{\epsilon}{4 n_{\mathsf{ind}}}$, output $\left(1, \frac{\mathsf{temp}_1}{N}, \frac{\mathsf{temp}_2}{N}\right)$ else output $(0, \perp, \perp)$.

Figure 7: Index Tracing

Algorithm $\mathsf{ID.Trace}(\mathsf{tk}, y, m_0, m_1, L, (i, p, q))$

Inputs: Tracing key tk, a parameter y, messages m_0, m_1, revocation list L, index i, and probabilities p, q.
Output : $\mathsf{id} \in \{0,1\}^\kappa$
Let $\epsilon = \lfloor 1/y \rfloor$. It sets $N = \lambda \cdot n_{\mathsf{ind}}/\epsilon$ and $\mathsf{temp}_\ell = 0$ for $\ell \in [\kappa]$. For $\ell = 1$ to κ, it does as follows:

1. For $j = 1$ to N
   It samples $b_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_j \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \ell, 0], m_{b_j}, L)$ and sends ct_j to D.
   If D outputs b_j, set $\mathsf{temp}_\ell = \mathsf{temp}_\ell + 1$, else set $\mathsf{temp}_\ell = \mathsf{temp}_\ell - 1$.

Let id be an empty string. For $\ell = 1$ to κ, do the following:

1. If $\frac{p+q}{2} > \frac{\mathsf{temp}_\ell}{N}$, set $\mathsf{id}_\ell = 0$, else set $\mathsf{id}_\ell = 1$.

Output $\mathsf{id} = \mathsf{id}_1 \| \cdots \| \mathsf{id}_\kappa$.

Figure 8: Identity Tracing

Efficiency. We can instantiate the above construction using public key and secret key RPE. The size of each parameter is directly inherited from that of the underlying RPE. We have $|x| = |\mathsf{id}| + \log n_{\mathsf{ind}} \leq |\mathsf{id}| + \lambda$ and $|f| := |f[j,\ell,b]| = \log n_{\mathsf{ind}} + \log \kappa + 1 = O(\lambda)$. We note that here, $|f[j,\ell,b]|$ refers to the description size of the function $|f[j,\ell,b]|$, not the size of the circuit implementing it. In particular, the latter can depend on $|\mathsf{id}|$ while the former is independent from $|\mathsf{id}|$.

Secret Tracing Setting. If we instantiate the scheme with our secret key RPE in Sec. 6, we have

$$|\mathsf{mpk}|, |\mathsf{ct}|, |\mathsf{sk}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|).$$

Public Tracing Setting. If we instantiate the scheme with our public key RPE in Sec. 4, we have

$$|\mathsf{mpk}|, |\mathsf{ct}| = \mathrm{poly}(\lambda, |f|, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{lb}|),\ \ |\mathsf{sk}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|).$$

8.2 Security

In this section we show that our construction of the indexed (secret/public tracing)-EITR scheme is secure.

IND-CPA security.

Theorem 8.2. *If (secret/public key)-*RPE *scheme satisfies (0-query-selective/adaptive) message hiding property (Def. 3.4/Def. 3.3) then our construction of indexed (secret/public tracing)-*EITR *scheme is (selective/adaptive)* IND-CPA *secure (Def. 7.2/Def. 7.3).*

Proof. Recall that for IND-CPA security, we need

$$\mathsf{Enc}(\mathsf{mpk}, m_0, L) \approx_c \mathsf{Enc}(\mathsf{mpk}, m_1, L).$$

For indexed *secret tracing* EITR scheme, this is equivalent to

$$\mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_0, L) \approx_c \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_1, L). \tag{8.1}$$

To prove this, we define the following hybrids:

Hybrid_0 : This is the real world with the challenge bit $b = 0$. In particular, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_0, L)$.

Hybrid_1 : In this hybrid, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0], m_0, L)$. Indistinguishability from Hybrid_0 follows from the broadcast security of secret-key RPE.

Hybrid_2 : In this hybrid, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0], m_1, L)$. Indistinguishability from Hybrid_1 follows from the message hiding property of secret-key RPE.

Hybrid_3 : In this hybrid, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_1, L)$. This is the real world with $b = 1$. Indistinguishability from Hybrid_2 follows from the broadcast security of secret-key RPE.

For indexed *public trace* EITR scheme, IND-CPA security is equivalent to

$$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0], m_0, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0], m_1, L). \tag{8.2}$$

The indistinguishability here follows directly from the message hiding property of the underlying RPE scheme. $\square$

Secure Tracing Analysis. We now show that our construction of indexed (secret/public tracing)-EITR scheme achieves the correct and false trace guarantees. The following analysis is mostly taken from ([GKW19], Section 5.2.2) with appropriate modifications to incorporate the revocation list.

False Trace Guarantee. The false trace guarantee of a trace and revoke scheme ensures that the tracing algorithm does not falsely accuse any user. We prove that there does not exist a PPT adversary who can output a decoder D such that the tracing algorithm, when executed using this decoder D, outputs an identity for which key was not queried by the adversary or the corresponding label is in the revocation list.

Theorem 8.3. *For every stateful* PPT *adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions* $\mathsf{negl}(\cdot)$*, such that for all $\lambda \in \mathbb{N}$*

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}(\lambda)$$

where the probability $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}$ *is as defined in Def. 7.5.*

Proof. Let $S \subseteq \mathcal{L} \times \mathcal{ID} \times [n_{\mathsf{ind}}]$ be the set of label-identity-index pairs on which $\mathcal{A}$ issues key queries. That is, $S = \{(\mathsf{lb}, \mathsf{id}, i) : \mathcal{A}$ issues a key query on $(\mathsf{lb}, \mathsf{id}, i)\}$. Let us first recall the assumption that $\mathcal{A}$ can issue at most one key query for any index $i \in [n_{\mathsf{ind}}]$. Next, we set up some notations. For $\mathsf{lb} \in \mathcal{L}$, $i \in [n_{\mathsf{ind}}]$, $\mathsf{id} \in \mathcal{ID}$ and a revocation list L, let

$$\begin{aligned}
S_{\mathsf{index}} &= \{i : (\mathsf{lb}, \mathsf{id}, i) \in S \text{ for some } (\mathsf{lb}, \mathsf{id}) \in \mathcal{L} \times \mathcal{ID}\}, \\
L_{\mathsf{index}} &= \{i : (\mathsf{lb}, \mathsf{id}, i) \in S \text{ and } \mathsf{lb} \in L\}, \\
B &= \{(\mathsf{id}, i) : (\mathsf{lb}, \mathsf{id}, i) \in S \text{ and } \mathsf{lb} \notin L\}, \\
B_{\mathsf{index}} &= \{i : (\mathsf{id}, i) \in B\}.
\end{aligned}$$

For any decoder box D, messages m_0, m_1, revocation list L, for any $i \in [n_{\mathsf{ind}} + 1], \ell \in [\kappa]$, $\mathsf{lb} \in \mathcal{L}$ we define

$$p^D_{i,\perp} = \Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \perp, 0], m_b, L)]$$

$$p^D_{i,\ell} = \Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \ell, 0], m_b, L)]$$

where the probability is taken over the random coins to decoder D and the randomness used during the encryption. We show that $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon(\lambda)} \leq \mathsf{negl}(\lambda)$. For $i \in [n_{\mathsf{ind}}], \ell \in [\kappa]$, we also define the following events:

$$\mathsf{Diff\text{-}Adv}^D_i : p^D_{i,\perp} - p^D_{i+1,\perp} > \epsilon/8n_{\mathsf{ind}}$$

$$\mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{lwr}} : p^D_{i,\perp} - p^D_{i,\ell} > \epsilon/16n_{\mathsf{ind}}$$

$$\mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{upr}} : p^D_{i,\ell} - p^D_{i+1,\perp} > \epsilon/16n_{\mathsf{ind}}$$

$$\mathsf{Diff\text{-}Adv}^D : \bigvee_{i \in [n_{\mathsf{ind}}] \setminus (S_{\mathsf{index}} \setminus L_{\mathsf{index}})} \mathsf{Diff\text{-}Adv}^D_i \bigvee_{\substack{(\mathsf{id},i) \in B, \ell \in [\kappa], \\ \text{s.t } \mathsf{id}_\ell = 1}} \mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{lwr}} \bigvee_{\substack{(\mathsf{id},i) \in B, \ell \in [\kappa] \\ \text{s.t } \mathsf{id}_\ell = 0}} \mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{upr}}$$

We will drop the dependence on D for the ease of notation. We have

$$\begin{aligned}\Pr[\mathsf{Fal\text{-}Tr}] &= \Pr[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}]\Pr[\overline{\mathsf{Diff\text{-}Adv}}] + \Pr[\mathsf{Fal\text{-}Tr} \mid \mathsf{Diff\text{-}Adv}]\Pr[\mathsf{Diff\text{-}Adv}] \\ &\leq \Pr[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] + \Pr[\mathsf{Diff\text{-}Adv}] \\ &= \Pr[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] + \sum_{i\in[n_{\mathsf{ind}}]} \Pr[i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_i] \\ &\quad + \sum_{(i,\ell)\in[n_{\mathsf{ind}}]\times[\kappa]} \Pr\Big[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (\mathsf{id}, i) \in B \\ &\qquad \wedge ([\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{lwr}} \wedge \mathsf{id}_\ell = 1] \vee [\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{upr}} \wedge \mathsf{id}_\ell = 0])\Big]\end{aligned}$$

Now, we argue that each of the terms on the RHS is bounded by a negligible function.

Lemma 8.4. *For every stateful* PPT *adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}_1(\cdot)$*, such that* $\forall\lambda \in \mathbb{N}$*, we have*

$$\Pr[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq \mathsf{negl}_1(\lambda).$$

Proof. Note that the event Fal-Tr occurs if and only if the trace algorithm outputs an identity that was not queried by the adversary or outputs an identity which was queried but also revoked. Since, the tracing scheme is two phased, the false tracing can happen in any of the two stages. First, during the index-tracing procedure, it can happen that $\exists(i, p, q) \in T_{\mathsf{index}}$, such that, $i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}}$ and secondly, during the identity-tracing procedure, the ID.Trace algorithm outputs an incorrect identity corresponding to some $i \in S_{\mathsf{index}} \backslash L_{\mathsf{index}}$. So, we have

$$\begin{aligned}&\Pr[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] \\ &\quad\leq \sum_{i\in[n_{\mathsf{ind}}]} \Pr[\mathsf{Fal\text{-}Tr} \wedge i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge (\exists p, q : (i, p, q) \in T_{\mathsf{index}}) \mid \overline{\mathsf{Diff\text{-}Adv}}] \\ &\quad + \sum_{(i,\ell)\in[n_{\mathsf{ind}}]\times[\kappa]} \Pr[\mathsf{Fal\text{-}Tr} \wedge \exists \mathsf{id}, \hat{\mathsf{id}} : (\mathsf{id}, i) \in B \wedge (\exists p, q : (i, p, q) \in T_{\mathsf{index}}) \\ &\qquad \wedge \hat{\mathsf{id}} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, 1^y, m_0, m_1, (i, p, q)) \wedge \mathsf{id}_\ell \neq \hat{\mathsf{id}}_\ell \mid \overline{\mathsf{Diff\text{-}Adv}}].\end{aligned}$$

Consider the first term on RHS. If $\overline{\mathsf{Diff\text{-}Adv}}$ happens then $\forall i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}}$ event $\overline{\mathsf{Diff\text{-}Adv}_i}$ happens.

We know that $\overline{\mathsf{Diff\text{-}Adv}_i}$ implies $p_{i,\perp} - p_{i+1,\perp} \leq \epsilon/8n_{\mathsf{ind}}$. Also the event $(\exists p, q : (i, p, q) \in T_{\mathsf{index}})$ implies $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} > \epsilon/4n_{\mathsf{ind}}$, where $\hat{p}_{j,\perp}$ is the estimated probability for $p_{j,\perp}$.

By applying Chernoff bound, we have, for every $i \in [n_{\mathsf{ind}}]$

$$\Pr[i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge (\exists p, q : (i, p, q) \in T_{\mathsf{index}}) \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq e^{-\lambda/24} \leq 2^{-O(\lambda)}.$$

Now, in the second term, for a fixed (i, ℓ) the probability term corresponds to the event when the ID.Trace outputs an identity $\hat{\mathsf{id}}$ corresponding to index i, such that $\hat{\mathsf{id}}_\ell \neq \mathsf{id}_\ell$, where id is such that the adversary made a key query for (id, i). So, if the event $\overline{\mathsf{Diff\text{-}Adv}}$ happens then $\forall(\mathsf{id}, i) \in B, \ell \in [\kappa]$, event $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,X}}$ happens where $X = \mathsf{lwr}$ if $\mathsf{id}_\ell = 1$ else $X = \mathsf{upr}$.

Thus, when we have $\mathsf{id}_\ell = 1$, then $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{lwr}}}$ implies $p_{i,\perp} - p_{i,\ell} \leq \epsilon/16n_{\mathsf{ind}}$.

Also, the event $(\exists\, p, q : (i, p, q) \in T_{\mathsf{index}})$ implies that $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} \geq \epsilon/4n_{\mathsf{ind}}$ and the event $\hat{\mathsf{id}} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, 1^y, m_0, m_1, (i, p, q)) \wedge \hat{\mathsf{id}}_\ell = 0$ implies $(\hat{p}_{i,\perp} + \hat{p}_{i+1,\perp} > 2\hat{p}_{i,\ell})$. These together imply that $\hat{p}_{i,\perp} - \hat{p}_{i,\ell} > \epsilon/8n_{\mathsf{ind}}$. Similarly, when $\mathsf{id}_\ell = 0$ and $\hat{\mathsf{id}}_\ell = 1$, $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{upr}}}$ implies

$p_{i,\ell} - p_{i+1,\perp} \leq \epsilon/16n_{\mathsf{ind}}$ and following the similar reasoning as above, we get $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} \geq \epsilon/4n_{\mathsf{ind}}$ and $\hat{p}_{i,\perp} + \hat{p}_{i+1,\perp} \leq 2\hat{p}_{i,\ell}$ which implies $\hat{p}_{i,\ell} - \hat{p}_{i+1,\perp} \geq \epsilon/8n_{\mathsf{ind}}$.
Hence, using Chernoff bound, we get that

$$\Pr[\exists \mathsf{id}, \hat{\mathsf{id}} : (\mathsf{id}, i) \in S \wedge (\exists p, q : (i, p, q) \in T_{\mathsf{index}}) \wedge \hat{\mathsf{id}} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, 1^y, m_0, m_1, (i, p, q)) \\ \wedge \hat{\mathsf{id}} \in T \wedge \mathsf{id}_\ell \neq \hat{\mathsf{id}}_\ell \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq 2^{-O(\lambda)}.$$

Combining the probability of both the RHS terms, we get

$$\Pr[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq n_{\mathsf{ind}} \cdot 2^{-O(\lambda)} + n_{\mathsf{ind}} \cdot \kappa \cdot 2^{-O(\lambda)} = \mathsf{negl}_1(\lambda).$$

□

Lemma 8.5. *If the underlying (secret/public key)-*RPE *satisfies (1-query-selective/adaptive) function hiding security property (Def. 3.6/Def. 3.5), then for every stateful* PPT *adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}_2(\cdot)$, *such that $\forall \lambda \in \mathbb{N}$ and $i \in [n_{\mathsf{ind}}]$, we have*

$$\Pr[i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_i] \leq \mathsf{negl}_2(\lambda).$$

Proof. We give a proof by contradiction. Suppose there exists a PPT adversary $\mathcal{A}$ that outputs a good decoder D along with messages m_0, m_1 and a revocation list L such that there exists $i^* \in [n_{\mathsf{ind}}]$ for which $\Pr[i^* \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}] \geq \delta$ where δ is some non-negligible function in the security parameter λ. We use this adversary $\mathcal{A}$ to build a reduction $\mathcal{B}$ that can break the function hiding security of the underlying RPE scheme. The reduction is as follows:

1. In the beginning, $\mathcal{A}$ outputs $1^{n_{\mathsf{ind}}}, 1^\kappa, L$ ($\mathcal{A}$ will output L adaptively in the public trae setting).

2. The RPE challenger samples $(\mathsf{RPE.mpk}, \mathsf{RPE.msk}) \leftarrow \mathsf{RPE.Setup}(1^\lambda)$ and sends $\mathsf{RPE.mpk}$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{mpk} = \mathsf{RPE.mpk}$ and forwards it to $\mathcal{A}$.

3. When $\mathcal{A}$ issues a secret key query $(\mathsf{lb}, \mathsf{id}, j)$ (recall that $\mathcal{A}$ can make at most one query for each index j), $\mathcal{B}$ sends a key query to the RPE challenger on $(\mathsf{lb}, x = (\mathsf{id}, j))$. The challenger returns $\mathsf{sk}_{\mathsf{lb},\mathsf{id},j}$ which $\mathcal{B}$ forwards to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a decoder D and messages m_0, m_1 (and a revocation list L in the case of public tracing EITR scheme) to $\mathcal{B}$. $\mathcal{B}$ then does the following:

   (a) Samples $i \leftarrow [n_{\mathsf{ind}}] \setminus (S_{\mathsf{index}} \setminus L_{\mathsf{index}})$ and sets $f_0 = f[i, \perp, 0]$ and $f_1 = f[i+1, \perp, 0]$.

   (b) Samples $b \leftarrow \{0,1\}$ and sends (f_0, f_1, m_b) to the RPE challenger. The challenger samples $\alpha \leftarrow \{0,1\}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_\alpha, m_b, L)$ and sends ct_1 to $\mathcal{B}$. (In public trace setting, $\mathsf{ct}_1 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_\alpha, m_b, L)$.)

   (c) $\mathcal{B}$ samples $\beta \leftarrow \{0,1\}$ and sends a encryption query on (f_β, m_b, L) and sets the ciphertext returned by the RPE challenger as ct_2. (In public trace setting, $\mathcal{B}$ computes ct_2 itself as $\mathsf{ct}_2 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_\beta, m_b, L)$.)

   (d) $\mathcal{B}$ samples $\beta \leftarrow \{0,1\}$, computes $\mathsf{ct}_2 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_\beta, m_b, L)$.

   (e) If $D(\mathsf{ct}_1) = D(\mathsf{ct}_2)$, sets $\alpha' = \beta$, else $\alpha' = 1 - \beta$.

   (f) $\mathcal{B}$ returns α' to the RPE challenger.

$\mathcal{B}$ wins the game if $\alpha = \alpha'$. Note that since $i \leftarrow [n_{\mathsf{ind}}] \setminus (S_{\mathsf{index}} \setminus L_{\mathsf{index}})$, for every key query $(\mathsf{lb}, \mathsf{id}, j)$ that $\mathcal{B}$ issues to the RPE challenger, either $f_0(\mathsf{id}, j) = f_1(\mathsf{id}, j)$ (when $i \notin S_{\mathsf{index}}$) or $\mathsf{lb} \in L$ (when $i \in S_{\mathsf{index}} \cap L_{\mathsf{index}}$). This establishes the admissibility of $\mathcal{B}$. Now, let us analyse the probability of $\mathcal{B}$ winning against the RPE challenger.

Let $q_{j,b} = \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[j, \perp, 0], m_b, L)]$.
Let E^D be the event that $\mathcal{B}$ wins and E_b^D be the event that $\mathcal{B}$ wins when it samples b as the message bit in step 4b. So, we have

$$\begin{aligned}
\mathsf{Pr}[\mathsf{E}_b^D] &= \frac{1}{4}\big(\mathsf{Pr}[\mathsf{E}_b^D \mid \alpha = 0, \beta = 0] + \mathsf{Pr}[\mathsf{E}_b^D \mid \alpha = 0, \beta = 1] \\
&\quad + \mathsf{Pr}[\mathsf{E}_b^D \mid \alpha = 1, \beta = 0] + \mathsf{Pr}[\mathsf{E}_b^D \mid \alpha = 1, \beta = 1]\big) \\
&= \frac{1}{4}\big(q_{i,b}^2 + (1 - q_{i,b})^2 + 2(q_{i,b}(1 - q_{i+1,b}) + (1 - q_{i,b})q_{i+1,b}) + q_{i+1,b}^2 + (1 - q_{i+1,b})^2\big) \\
&= \frac{1}{2} + \frac{(q_{i,b} - q_{i+1,b})^2}{2}.
\end{aligned}$$

We have assumed $\exists i^* \in [n_{\mathsf{ind}}]$ such that $\mathsf{Pr}[i^* \notin S_{\mathsf{index}} \setminus L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}] \geq \delta$. $\mathsf{Pr}[i = i^*] = 1/n_{\mathsf{ind}}$. So we get $\mathsf{Pr}[i = i^* \wedge i^* \notin S_{\mathsf{index}} \setminus L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}] \geq \delta / n_{\mathsf{ind}}$.

Let F be the event: $\exists i^* \in [n_{\mathsf{ind}}]$ such that $i = i^* \wedge i^* \notin S_{\mathsf{index}} \setminus L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}$. Then when F occurs, we have $p_{i,\perp} - p_{i+1,\perp} > \epsilon / 8n_{\mathsf{ind}} \Rightarrow \exists b' \in \{0,1\}$ s.t $q_{i,b'} - q_{i+1,b'} > \epsilon / 8n_{\mathsf{ind}}$. Also, $\mathsf{Pr}[\mathsf{E}_b^D] \geq 1/2$ for $b \in \{0,1\}$, irrespective of occurrence of F. Now,

$$\begin{aligned}
\mathsf{Pr}[\mathsf{E}_{b'}^D] &= \mathsf{Pr}[\mathsf{E}_{b'}^D \mid \mathsf{F}]\mathsf{Pr}[\mathsf{F}] + \mathsf{Pr}[\mathsf{E}_{b'}^D \mid \bar{\mathsf{F}}]\mathsf{Pr}[\bar{\mathsf{F}}] \\
&\geq (1/2 + \frac{\epsilon^2}{128 {n_{\mathsf{ind}}}^2}) \times (\delta / n_{\mathsf{ind}}) + 1/2 \times (1 - \delta / n_{\mathsf{ind}}) \\
&= 1/2 + \frac{\epsilon^2 \delta}{128 {n_{\mathsf{ind}}}^3}.
\end{aligned}$$

Again,

$$\mathsf{Pr}[\mathsf{E}^D] = \frac{\mathsf{Pr}[\mathsf{E}_{b'}^D]}{2} + \frac{\mathsf{Pr}[\mathsf{E}_{\bar{b}'}^D]}{2} \geq \frac{1}{2}\left(\frac{1}{2} + \frac{\epsilon^2 \delta}{128 {n_{\mathsf{ind}}}^3}\right) + \frac{1}{4} = \frac{1}{2} + \eta, \text{ where } \eta = \frac{\epsilon^2 \delta}{128 {n_{\mathsf{ind}}}^3}.$$

Thus, $\mathcal{B}$ wins against the function-hiding security of the underlying RPE scheme with advantage $\geq \eta$, which is non-negligible for non-negligible ϵ and δ, a contradiction. Hence the lemma follows.

□

Lemma 8.6. *If the underlying (secret/public key)-*RPE *scheme satisfies (1-query-selective/adaptive) function hiding security property (Def. 3.6/Def. 3.5), then for every stateful* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}_3(\cdot)$*, such that* $\forall \lambda \in \mathbb{N}$ *and* $i \in [n_{\mathsf{ind}}], \ell \in [\kappa]$*, we have*

$$\mathsf{Pr}\Big[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (\mathsf{id}, i) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{lwr}} \wedge \mathsf{id}_\ell = 1] \vee [\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{upr}} \wedge \mathsf{id}_\ell = 0])\Big] \leq \mathsf{negl}_3(\lambda).$$

Proof. The proof is similar to the proof of Lemma 8.5. We show that if there exists a PPT adversary $\mathcal{A}$ who outputs a good decoder D along with messages m_0, m_1 and a revocation list L for which there exists $i^* \in [n_{\mathsf{ind}}], \ell^* \in [\kappa]$ such that $\mathsf{Pr}[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (i^*, \mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*} = 1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*} = 0])] \geq \delta$.

We use this adversary $\mathcal{A}$ to build a reduction $\mathcal{B}$ that can break the function hiding security of the underlying RPE scheme. $\mathcal{B}$ is defined in the same way as in the proof of Lemma 8.5, except Step 4a, which is now described as follows:

$\mathcal{B}$ samples $i \leftarrow B_{\mathsf{index}}$, $\ell \in [\kappa]$ and a bit $g \leftarrow \{0,1\}$ and sets $f_0 = f[i,\perp,0]$ and $f_1 = f[i,\ell,0]$, if $g = 0$, else sets $f_0 = f[i,\ell,0]$ and $f_1 = f[i+1,\perp,0]$.

Analysis of $\mathcal{B}$'s advantage:
Let $q_{j,k,b} = \Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[j,k,0], m_b, L)]$.
Let E^D be the event that $\mathcal{B}$ wins and E_b^D be the event that $\mathcal{B}$ wins when $\mathcal{B}$ samples b as the message bit in step 4b. So, we have

$$\begin{aligned}
\Pr[\mathsf{E}_b^D] = & \frac{1}{8}\big(\Pr[\mathsf{E}_b^D \mid \alpha = 0, \beta = 0, g = 0] + \Pr[\mathsf{E}_b^D \mid \alpha = 0, \beta = 1, g = 0] \\
& +\Pr[\mathsf{E}_b^D \mid \alpha = 1, \beta = 0, g = 0] + \Pr[\mathsf{E}_b^D \mid \alpha = 1, \beta = 1, g = 0] \\
& +\Pr[\mathsf{E}_b^D \mid \alpha = 0, \beta = 0, g = 1] + \Pr[\mathsf{E}_b^D \mid \alpha = 0, \beta = 1, g = 1] \\
& +\Pr[\mathsf{E}_b^D \mid \alpha = 1, \beta = 0, g = 1] + \Pr[\mathsf{E}_b^D \mid \alpha = 1, \beta = 1, g = 1]\big) \\
= & \frac{1}{8}\big(q_{i,\perp,b}^2 + (1-q_{i,\perp,b})^2 + 2(q_{i,\perp,b}(1-q_{i,\ell,b}) + (1-q_{i,\perp,b})q_{i,\ell,b}) + q_{i,\ell,b}^2 + \\
& (1-q_{i,\ell,b})^2 + q_{i,\ell,b}^2 + (1-q_{i,\ell,b})^2 + 2(q_{i,\ell,b}(1-q_{i+1,\perp,b}) + \\
& (1-q_{i,\ell,b})q_{i+1,\perp,b}) + q_{i+1,\perp,b}^2 + (1-q_{i+1,\perp,b})^2\big) \\
= & \frac{1}{2} + \frac{(q_{i,\perp,b} - q_{i,\ell,b})^2}{4} + \frac{(q_{i,\ell,b} - q_{i+1,\perp,b})^2}{4}.
\end{aligned}$$

We have assumed that $\exists\ i^* \in [n_{\mathsf{ind}}], \ell^* \in [\kappa]$ such that $\Pr[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (i^*, \mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*} = 1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*} = 0])] \geq \delta$.
Using $\Pr[i = i^* \wedge \ell = \ell^*] = 1/\kappa n_{\mathsf{ind}}$, we get $\Pr[(i = i^* \wedge \ell = \ell^*) \wedge \exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } ((i^*, \mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*} = 1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*} = 0]))] \geq \frac{\delta}{\kappa n_{\mathsf{ind}}}$
Let F be the event: $\exists\ i^* \in [n_{\mathsf{ind}}], \ell^* \in [\kappa]$ such that $(\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (i^*, \mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*} = 1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*} = 0]))$. Then when F occurs, we have $(p_{i,\perp} - p_{i,\ell} > \epsilon/8n_{\mathsf{ind}}) \vee (p_{i,\ell} - p_{i+1,\perp} > \epsilon/8n_{\mathsf{ind}})$. This implies that there exists $b' \in \{0,1\}$ s.t $(q_{i,\perp,b'} - q_{i,\ell,b'} > \epsilon/8n_{\mathsf{ind}}) \vee ((q_{i,\ell,b'} - q_{i+1,\perp,b'} > \epsilon/8n_{\mathsf{ind}}))$. Also, $\Pr[\mathsf{E}_b^D] \geq 1/2$ for $b \in \{0,1\}$, irrespective of occurrence of F. Now,

$$\begin{aligned}
\Pr[\mathsf{E}_{b'}^D] = & \Pr[\mathsf{E}_{b'}^D \mid \mathsf{F}]\Pr[\mathsf{F}] + \Pr[\mathsf{E}_{b'}^D \mid \bar{\mathsf{F}}]\Pr[\bar{\mathsf{F}}] \\
\geq & (1/2 + \frac{\epsilon^2}{256 {n_{\mathsf{ind}}}^2}) \times (\delta/\kappa n_{\mathsf{ind}}) + 1/2 \times (1 - \delta/\kappa n_{\mathsf{ind}}) \\
= & 1/2 + \frac{\epsilon^2 \delta}{256 \kappa {n_{\mathsf{ind}}}^3}.
\end{aligned}$$

Again,

$$\Pr[\mathsf{E}^D] = \frac{\Pr[\mathsf{E}_{b'}^D]}{2} + \frac{\Pr[\mathsf{E}_{\bar{b'}}^D]}{2} \geq \frac{1}{2}\left(\frac{1}{2} + \frac{\epsilon^2 \delta}{256 \kappa {n_{\mathsf{ind}}}^3}\right) + \frac{1}{4} = \frac{1}{2} + \eta, \text{ where } \eta = \frac{\epsilon^2 \delta}{512 {n_{\mathsf{ind}}}^3 \kappa}.$$

□

Combining the result of Lemmas 8.4, 8.5 and 8.6, we get

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda) + n_{\mathsf{ind}} \cdot \mathsf{negl}_2(\lambda) + n_{\mathsf{ind}} \cdot \kappa \cdot \mathsf{negl}_3(\lambda) = \mathsf{negl}(\lambda).$$

□

Correct trace guarantee. We prove that whenever an adversary outputs a good decoder, the tracing algorithm will output, with all but negligible probability, at least one valid user identity which was queried by the adversary.

Theorem 8.7. *If the underlying (secret/public key)-*RPE *scheme in the indexed (secret/public tracing)-*EITR *scheme construction satisfies (1-query-selective broadcast and message/adaptive message) hiding property, then for every stateful* PPT *adversary $\mathcal{A}$ for the (selective/adaptive)-tracing game (Def. 7.6/Def. 7.5) and non-negligible function ϵ, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$, the following holds*

$$\Pr[\mathsf{Cor\text{-}Tr}] \geq \Pr[\mathsf{Good\text{-}Decoder}] - \mathsf{negl}(\lambda)$$

Proof. Let us define $p^D_{\mathsf{Broadcast}} = \Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_b, L)]$. Then in the secret trace setting, if the event $\mathsf{Good\text{-}Decoder}_{\mathcal{A},\epsilon}$ occurs, then this implies $p^D_{\mathsf{Broadcast}} \geq 1/2 + \epsilon$. Further, from the broadcast security of (secret-key) RPE, we get $p^D_{1,\perp} \geq p^D_{\mathsf{Broadcast}} - \mathsf{negl}_1(\lambda)$, which implies

$$p^D_{1,\perp} \geq 1/2 + \epsilon - \mathsf{negl}_1(\lambda). \tag{8.3}$$

In public trace setting, the event $\mathsf{Good\text{-}Decoder}_{\mathcal{A},\epsilon}$ directly implies $p^D_{1,\perp} \geq 1/2+\epsilon$ by definition. Also, by message hiding property of the underlying (secret/public key) RPE scheme, we have

$$p^D_{n_{\mathsf{ind}}+1,\perp} \leq 1/2 + \mathsf{negl}_2(\lambda) \tag{8.4}$$

for some negligible function $\mathsf{negl}_2(\cdot)$ with overwhelming probability. This is so, because $f[n_{\mathsf{ind}}, \perp, 0](\mathsf{id}, i) = 0$ for all $(\mathsf{id}, i) \in \mathcal{ID} \times [n_{\mathsf{ind}}]$.
Combining equations (8.3) and (8.4), we get $p^D_{1,\perp} - p^D_{n_{\mathsf{ind}}+1,\perp} > \epsilon/2$.

Let $S^{\mathsf{ind}} = \{i \in [n_{\mathsf{ind}}] \mid p^D_{i,\perp} - p^D_{i+1,\perp} > \epsilon/2n_{\mathsf{ind}}\}$. Then if the event Good-Decoder occurs, $S^{\mathsf{ind}} \neq \phi$.

By Chernoff bound, we have

$$\forall i \in S^{\mathsf{index}}, \quad \Pr\left[\hat{p}^D_{i,\perp} - \hat{p}^D_{i+1,\perp} \leq \epsilon/4n_{\mathsf{ind}}\right] \leq 2^{-O(\lambda)} = \mathsf{negl}_3(\lambda)$$

for some negligible function $\mathsf{negl}_3(\cdot)$. Here, $\hat{p}$ denotes the estimate for p computed by tracing algorithm.

So, with all but negligible probability,

$$T_{\mathsf{index}} \neq \phi \text{ and } \forall (i,p,q) \in T_{\mathsf{index}}, p - q > \epsilon/4n_{\mathsf{ind}}$$

where T_{index} and p, q are as defined in the Trace algorithm 8.1.
Note that the ID.Trace algorithm (Figure 8) takes as input $(i,p,q) \in T_{\mathsf{index}}$ and outputs a corresponding id, where $\mathsf{id}_\ell = 1$ if $\hat{p}^D_{i,\ell} > (p+q)/2$ else $\mathsf{id}_\ell = 0$ for $\ell \in [\kappa]$. Then, for every $(i,p,q) \in T_{\mathsf{index}}$, the tracing algorithm outputs an identity. Hence, if $T_{\mathsf{index}} \neq \phi \Rightarrow T \neq \phi$, where T is defined as in the tracing algorithm. So,

$$\begin{aligned}\Pr[T \neq \phi] \geq & \Pr[T \neq \phi \wedge \mathsf{Good\text{-}Decoder}] \\ \geq & (1 - \mathsf{negl}_2(\lambda)) \Pr[\mathsf{Good\text{-}Decoder}] \\ \geq & \Pr[\mathsf{Good\text{-}Decoder}] - \mathsf{negl}(\lambda)\end{aligned}$$

for some negligible function $\mathsf{negl}(\cdot)$. Combining this with the false trace guarantee, we have

$$\Pr[\mathsf{Cor\text{-}Tr}] \geq \Pr[\mathsf{Good\text{-}Decoder}] - \mathsf{negl}(\lambda).$$

□

9 Bounded Trace and Revoke with Embedded identity

In this section we show how to construct a bounded (secret/public tracing)-EITR scheme from an indexed (secret/public tracing)-EITR scheme. The construction and security analysis in this section is an adaptation of ([GKW19], Section 9) with appropriate modifications to incorporate the revocation list L. We present the construction and proofs for the secret trace setting primarily and also outline the differences in the public trace setting simultaneously.

9.1 Construction

Let $\mathsf{Ind\text{-}TR} = (\mathsf{Ind.Setup}, \mathsf{Ind.KeyGen}, \mathsf{Ind.Enc}, \mathsf{Ind.Dec}, \mathsf{Ind.Trace})$ be an indexed (secret/public tracing)-EITR system. Let $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ be a signature scheme that satisfies unforgeability with signature space $\{0,1\}^{\ell_s}$. We let n_{bd} denote the bound on the number of key queries that the adversary can make. For our purpose, we can assume $n_{\mathsf{bd}} \leq 2^{\lambda}$. We construct a bounded (secret/public tracing)-EITR scheme as follows:

$\mathsf{Setup}(1^{\lambda}, 1^{\kappa}, n_{\mathsf{bd}}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following:

1. Set $n_{\mathsf{index}} = 2{n_{\mathsf{bd}}}^2$.
2. For $j = 1$ to λ, sample $(\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j) \leftarrow \mathsf{Ind.Setup}(1^{\lambda}, 1^{\kappa'}, n_{\mathsf{index}})$, where κ' is $\kappa + \ell_s$.
3. Sample $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^{\lambda})$.
4. Output $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]})$ and $\mathsf{msk} = (\mathsf{sig.vk}, \mathsf{sig.sk},$ $\{\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j\}_{j\in[\lambda]})$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{lb},\mathsf{id}}$. The key generation algorithm does the following:

1. Parse msk as $(\mathsf{sig.vk}, \mathsf{sig.sk}, \{\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j\}_{j\in[\lambda]})$.
2. Compute $\sigma = \mathsf{Sig.Sign}(\mathsf{sig.sk}, \mathsf{id})$ and let $\mathsf{id}' = (\mathsf{id}, \sigma)$.
3. For $j = 1$ to λ, do the following:
   (a) Sample $i_j \leftarrow [2{n_{\mathsf{bd}}}^2]$.
   (b) $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}_j, \mathsf{lb}, \mathsf{id}', i_j)$.
4. Return $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}\}_{j\in[\lambda]}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. The encryption algorithm does the following:

1. Parse mpk as $(\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]})$.
2. For $j = 1$ to $\lambda - 1$, randomly sample $r_j \leftarrow \mathcal{M}$ and set $r_{\lambda} = m \oplus r_1 \oplus \ldots \oplus r_{\lambda-1}$.
3. For $j = 1$ to λ, compute $\mathsf{Ind.ct}_j \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_j, r_j, L)$.
4. Return $\mathsf{ct} = \{\mathsf{Ind.ct}_j\}_{j\in[\lambda]}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{id}}, \mathsf{ct}, L) \to m'$. The decryption algorithm does the following:

1. Parse $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}\}_{j\in[\lambda]}$ and $\mathsf{ct} = \{\mathsf{Ind.ct}_j\}_{j\in[\lambda]}$.
2. For $j = 1$ to λ, compute $r'_j = \mathsf{Ind.Dec}(\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}, \mathsf{Ind.ct}_j, L)$.
3. If any of the decryption fails then output $\perp$, else output $m' = r'_1 \oplus \cdots \oplus r'_{\lambda}$.

$\mathsf{Trace}^D(\mathsf{tk}, y, m_0, m_1, L) \to T^{\mathsf{final}}$. The trace algorithm uses two algorithms Bnd-isGoodDecoder and Bnd-Subtrace defined in Figures 9 and 10, respectively as subroutines and is defined as follows:

1. Parse tk as $\mathsf{msk} = (\mathsf{sig.vk}, \mathsf{sig.sk}, \{\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j\}_{j\in[\lambda]})$ and let $\mathsf{Ind.tk}_j = \mathsf{Ind.msk}_j$ for $j \in [\lambda]$. (In the public trace setting, $\mathsf{tk} = \mathsf{mpk}$ and $\mathsf{Ind.tk}_j = \mathsf{Ind.mpk}_j$).
2. Set $j = 1$.
3. Set $\mathsf{flag} = 0$. For $\mathsf{itr} = 1$ to $\lambda \cdot y$, do the following
   (a) Choose a random message $r \leftarrow \mathcal{M}$.
   (b) Run Bnd-isGoodDecoder as $\mathsf{flag} \leftarrow \mathsf{Bnd\text{-}isGoodDecoder}^D(\{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$.
   (c) If $\mathsf{flag} = 1$, break. Else, continue.
4. If $\mathsf{flag} = 1$, run Bnd-Subtrace as $T \leftarrow \mathsf{Bnd\text{-}Subtrace}^D(\{\mathsf{Ind.mpk}_j, \mathsf{Ind.tk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$. Else, set $T = \phi$.
5. If $T = \phi$ and $j < \lambda$, set $j = j + 1$ and go to step 3. Otherwise do the following:
   - Set $T^{\mathsf{temp}} = \phi$. For each $\mathsf{id}' = (\mathsf{id}, \sigma) \in T$, if $\mathsf{Sig.Verify}(\mathsf{sig.vk}, \mathsf{id}, \sigma) = 1$, add id to T^{temp}. Concretely,
   $$T^{\mathsf{temp}} = \{\mathsf{id} : \exists \sigma\ s.t.\ (\mathsf{id}, \sigma) \in T \text{ and } \mathsf{Sig.Verify}(\mathsf{sig.vk}, \mathsf{id}, \sigma) = 1\}.$$
   - Recall the function $\mathsf{map} : \mathcal{ID} \to \mathcal{L}$, that maps a given identity id to its corresponding label lb (Remark 7.4). For each $\mathsf{id} \in T^{\mathsf{temp}}$, if $\mathsf{map}(\mathsf{id}) \in L$, then set $T^{\mathsf{temp}} = T^{\mathsf{temp}} \setminus \{\mathsf{id}\}$.
   - Set $T^{\mathsf{final}} = T^{\mathsf{temp}}$.
   - If $T^{\mathsf{final}} = \phi$ and $j < \lambda$, set $j = j + 1$ and go to step 3. Otherwise exit and return T^{final}.

Correctness. We prove that the above construction of bounded (secret/public tracing)-EITR scheme satisfies correctness (Def. 7.1) via the following theorem.

Theorem 9.1. *Assume* Ind-TR *is a correct indexed (secret/public tracing)*-EITR *scheme then the above construction of bounded (secret/public tracing)*-EITR *scheme is correct.*

Proof. We have, as per the construction, that any message m is split in λ components $r_1, \ldots, r_\lambda$ such that $r_1 \oplus \cdots \oplus r_\lambda = m$. Then, from the correctness of Ind.TR, we get $r'_k = r_k$ for all $k \in [\lambda]$, where $r'_k \leftarrow \mathsf{Ind.Dec}(\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},k}, \mathsf{Ind.ct}_k, L)$, as long as $\mathsf{lb} \notin L$. Hence, the decryption algorithm correctly outputs m as $r_1 \oplus \cdots \oplus r_\lambda$. □

Efficiency. We can instantiate the above construction by the indexed public/secret tracing-EITR scheme in Sec. 8.1. The above construction is basically λ times repetition of the underlying indexed EITR scheme. Additional overhead is induced due to the usage of the signature scheme, where identity becomes longer by $\ell_s = \mathrm{poly}(\lambda)$ bit and the master public key is longer by $|\mathsf{sig.vk}| = \mathrm{poly}(\lambda)$ bit.[26] These changes do not alter the dependency on $|\mathsf{id}|$ and $|\mathsf{lb}|$ of the parameter size. Therefore, the parameter size is as follows.

[26]Parameter sizes of most signature schemes depend on the length of the message space, which is $|\mathsf{id}|$ in our case. However, this dependency can be removed by hashing the message before signing using the collision resistant hash functions.

Algorithm $\mathsf{Bnd\text{-}isGoodDecoder}^D(\mathsf{key}, 1^y, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{Ind.mpk}_j\}_{j \in [\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and position-index $i \in [\lambda]$.
Output : $0/1$.

1. Set $\mathsf{count} = 0$. Let $\epsilon = 1/y$.
2. For $j = 1$ to $\lambda \cdot y$:
   - Split r in $\lambda - 1$ shares. That is, randomly sample $\lambda - 1$ messages, $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_\lambda$ such that $\oplus_{k \in [\lambda] \setminus \{i\}} r_k = r$.
   - Sample $b \leftarrow \{0, 1\}$. For $k \in [\lambda] \setminus \{i\}$, compute $\mathsf{Ind.ct}_k \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_k, r_k, L)$. Compute $\mathsf{Ind.ct}_i \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_i, r \oplus m_b, L)$.
   - Query D on $\mathsf{ct} = (\mathsf{Ind.ct}_1, \ldots, \mathsf{Ind.ct}_\lambda)$. Let b' be D's response.
   - If $b' = b$, set $\mathsf{count} = \mathsf{count} + 1$.
3. If $\mathsf{count}/(\lambda \cdot y) \geq 1/2 + \epsilon/3$, then output 1, else output 0.

Figure 9: Algorithm Bnd-isGoodDecoder

Algorithm $\mathsf{Bnd\text{-}Subtrace}^D(\mathsf{key}, 1^y, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{Ind.mpk}_j, \mathsf{Ind.tk}_j\}_{j \in [\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and index-position $i \in [\lambda]$.
Output : $T \subseteq \{0, 1\}^\kappa$

1. Define oracle $\tilde{D}[\{\mathsf{Ind.mpk}_j\}_{j \in [\lambda]}, r, L, i]$ as in Figure 11.
2. Output $T \leftarrow \mathsf{Ind.Trace}^{\tilde{D}}(\mathsf{Ind.tk}_i, 4y, m_0 \oplus r, m_1 \oplus r, L)$.

Figure 10: Algorithm Bnd-Subtrace

Algorithm $\tilde{D}^D[\mathsf{key}, r, L, i]$

Hardwired values: keys $\mathsf{key} = \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}$, message r, revocation list L and index-position $i \in [\lambda]$.
Inputs: $\mathsf{Ind.ct}$.
Output : 0/1
Upon input $\mathsf{Ind.ct}$, the $\tilde{D}$ oracle does the following:

- Shares r in $\lambda - 1$ components as follows: it chooses $\lambda - 1$ random messages r_k for $k \in [\lambda] \setminus \{i\}$, such that $\oplus_{k\in[\lambda]\setminus\{i\}} r_k = r$.
- For $k \in [\lambda] \setminus \{i\}$, computes $\mathsf{Ind.ct}_k = \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_k, r_k, L)$.
- Sets $\mathsf{ct}_{\mathsf{bd}} = (\mathsf{Ind.ct}_1, \ldots, \mathsf{Ind.ct}_{i-1}, \mathsf{Ind.ct}, \mathsf{Ind.ct}_{i+1}, \ldots, \mathsf{Ind.ct}_\lambda)$.
- Queries oracle D as $b' \leftarrow D(\mathsf{ct}_{\mathsf{bd}})$.
- Outputs b'.

Figure 11: Oracle $\tilde{D}$

Secret Tracing Setting. In the secret tracing setting, we have

$$|\mathsf{mpk}|, |\mathsf{ct}|, |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|).$$

Public Tracing Setting. In the public tracing setting, we have

$$|\mathsf{mpk}|, |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathsf{lb}|), \ |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|).$$

9.2 Security

In this section, we prove that our construction of bounded (secret/public tracing)-EITR scheme is secure.

IND-CPA security.

Theorem 9.2. *If* Ind-TR *is a (selective/adaptive)*-IND-CPA *secure indexed (secret/public tracing)*-EITR *scheme, then the above construction of bounded (secret/public tracing)*-EITR *scheme is (selective/adaptive)*IND-CPA *secure.*

Proof. We show that if there exists a PPT adversary that breaks the IND-CPA security of the bounded EITR scheme, then we can use $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that breaks IND-CPA security of the underlying Ind-TR scheme. □

The reduction is as follows:

1. $\mathcal{B}$ gets $(1^\kappa, 1^{n_{\mathsf{bd}}}, L)$ from the adversary $\mathcal{A}$. It sets $n_{\mathsf{index}} = 2{n_{\mathsf{bd}}}^2$ and sends $(1^{\kappa+\ell_s}, 1^{n_{\mathsf{index}}}, L)$ to the Ind-TR challenger. The Ind-TR challenger returns $\mathsf{Ind.mpk}$. (In the public trace setting, $\mathcal{A}$ outputs L adaptively, along with the challenge messages in Step 4).

2. $\mathcal{B}$ sets $\mathsf{Ind.mpk}_1 = \mathsf{Ind.mpk}$, generates $(\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j) \leftarrow \mathsf{Ind.Setup}(1^\lambda, 1^{\kappa+\ell_s}, n_{\mathsf{index}})$ for $j \in \{2, \cdots, \lambda\}$ and $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. It sends $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ makes a key query $(\mathsf{lb}, \mathsf{id})$, $\mathcal{B}$ samples $i_1 \leftarrow [n_{\mathsf{index}}]$, computes id' as in the construction and sends a key query $(\mathsf{lb}, \mathsf{id}', i_1)$ to the Ind-TR challenger. The Ind-TR challenger returns $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id}} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}, \mathsf{lb}, \mathsf{id}', i_1)$. $\mathcal{B}$ sets $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},1} = \mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id}}$, computes $\mathsf{sk}_{\mathsf{lb},\mathsf{id},j}$ for $j \in \{2, \cdots, \lambda\}$ itself and returns $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}\}_{j\in\lambda}$ to $\mathcal{A}$.

4. When $\mathcal{A}$ sends the challenge messages (m_0, m_1) for the challenge query, $\mathcal{B}$ samples $r_j \leftarrow \mathcal{M}$ for $j \in \{2, \cdots, \lambda\}$, sets $m'_b = \bigoplus_{j>1} r_j \oplus m_b$, sends (m'_0, m'_1) for the challenge query to the Ind-TR challenger and gets back Ind.ct. $\mathcal{B}$ sets $\mathsf{ct}_1 = \mathsf{Ind.ct}$, computes $\mathsf{ct}_j \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_j, r_j, L)$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{ct} = \{\mathsf{ct}_j\}_{j\in\lambda}$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ returns b' to the challenger.

Observe that $\mathcal{B}$ issues key queries to the Ind-TR challenger only when there is a key query from $\mathcal{A}$. From the admissibility of $\mathcal{A}$, for any key query of the form $(\mathsf{lb}, \mathsf{id}, i_1)$ that $\mathcal{B}$ issues $\mathsf{lb} \in L$. So, $\mathcal{A}$ wins the IND-CPA security game of the BD-TR scheme with advantage ϵ, then $\mathcal{B}$ also wins the IND-CPA security game of the underlying Ind-TR scheme with the same advantage.

Secure Tracing Analysis. Now we show that our construction satisfies false trace and correct trace guarantees.

False Trace Guarantee. First, we show that the probability of false trace in our scheme is negligible in the security parameter via the following theorem.

Theorem 9.3. *Assume that* Sig *is unforgeable, then our construction of bounded (secret/public tracing)-* EITR *scheme satisfies the (selective/adaptive) false trace guarantee (Def. 7.8/Def. 7.9), even if the adversary makes unbounded polynomial number of key queries.*

Proof. Let us first setup some notations. For $\mathsf{lb} \in \mathcal{L}$, $\mathsf{id} \in \mathcal{ID}$ and a revocation list L, let

$$\begin{aligned} S_{\mathcal{ID}} &= \{\mathsf{id} : (\mathsf{lb}, \mathsf{id}) \in S\}, \\ L_{\mathcal{ID}} &= \{\mathsf{id} : (\mathsf{lb}, \mathsf{id}) \in S \wedge \mathsf{lb} \in L\},, \\ T^{\mathsf{final}}_{\mathsf{lb}} &= \{\mathsf{map}(\mathsf{id}) : \mathsf{id} \in T^{\mathsf{final}}\} \end{aligned}$$

where map is as defined in Remark 7.4 and T^{final} is as in the construcion.
False trace happens when $T^{\mathsf{final}} \not\subseteq S_{\mathcal{ID}}$ or $T^{\mathsf{final}}_{\mathsf{lb}} \cap L \neq \phi$. Observe that $T^{\mathsf{final}}_{\mathsf{lb}} \cap L = \phi$ by definition. So, all we need to argue is that $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}}$.

Recall that the tracing algorithm uses Bnd-Subtrace algorithm to find a set $T = \{(\mathsf{id}_k, \sigma_k)\}_k$. Identity id_k is added to the set T^{temp} and then to set T^{final} only if $\mathsf{Sig.Verify}(\mathsf{sig.vk}, \mathsf{id}_k, \sigma_k) = 1$. Next we show that if there is an adversary $\mathcal{A}$ who outputs a decoder D along with messages m_0, m_1 and revocation list L such that the tracing algorithm outputs T^{final} where $T^{\mathsf{final}} \not\subseteq S_{\mathcal{ID}}$, then there exists a reduction $\mathcal{B}$ against unforgeability of signature scheme Sig. The reduction is defined as follows:

Upon receiving sig.vk from the Sig challenger, $\mathcal{B}$ does the following:

1. Run $\mathcal{A}$ on input 1^λ to obtain $(1^\kappa, 1^{n_{\mathsf{index}}}, L)$. (In the public trace setting, it outputs L adaptively along with the challenge messages).

2. Samples $(\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j) \leftarrow \mathsf{Ind.Setup}(1^\lambda, 1^{\kappa'}, n_{\mathsf{index}})$ for $j \in [\lambda]$ and sends $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{mpk}_j\}_{j\in[\lambda]})$ to $\mathcal{A}$.

3. Whenever $\mathcal{A}$ issues a key query $(\mathsf{lb}, \mathsf{id})$, $\mathcal{B}$ sends id to the Sig challenger for signature. The Sig challenger returns $\sigma = \mathsf{Sig}(\mathsf{sig.sk}, \mathsf{id})$. $\mathcal{B}$ generates the secret key $\mathsf{sk}_{\mathsf{lb},\mathsf{id}}$ using $\mathsf{id}' = (\mathsf{id}, \sigma)$ as in the construction and sends $\mathsf{sk}_{\mathsf{lb},\mathsf{id}}$ to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a decoder D, messages m_0, m_1.

5. $\mathcal{B}$ runs the trace algorithm with the help of the decoder D and gets a set T^{final}.

6. If there exists an $\mathsf{id}^* \in T^{\mathsf{final}}$ such that $\mathsf{id}^* \notin S_{\mathcal{ID}}$, then $\mathcal{B}$ returns a forgery for id^* to Sig challenger as follows: $\mathsf{id}^* \in T^{\mathsf{final}}$ implies that there exists σ^*, such that $(\mathsf{id}^*, \sigma^*) \in T$, and $\mathsf{Sig.Verify}(\mathsf{sig.vk}, \mathsf{id}^*, \sigma^*) = 1$. $\mathcal{B}$ returns $(\mathsf{id}^*, \sigma^*)$ as a forgery to the Sig challenger.

   Note that since $\mathsf{id}^* \notin S_{\mathcal{ID}}$, $\mathcal{B}$ must not have queried a signature on id^* to the Sig challenger and hence $(\mathsf{id}^*, \sigma^*)$ is a valid forgery. If the false tracing happens with non-negligible probability ϵ, then $\mathcal{B}$ also wins the unforgeability game with probability ϵ.

□

Correct Trace Guarantee. Recall that in the experiment for correct tracing, the adversary first sends $(1^\kappa, 1^{n_{\mathsf{bd}}})$ and a revocation list L (in the public trace setting, it outputs L adaptively), then the challenger sends the public key to the adversary. After that the adversary is allowed to make at most n_{bd} key queries. Let $S = \{(\mathsf{lb}, \mathsf{id})\}$ be the set of label-identity pairs for which the adversary issues key queries. At the end, the adversary outputs a decoder box D along with two messages m_0 and m_1. If D is a good decoder then for correct tracing we want that the tracing algorithm outputs non empty set of traitors $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$. We prove the correct trace guarantee of our scheme via the following theorem.

Theorem 9.4. *Assume that the underlying (secret/public tracing)-*Ind-TR *scheme satisfies (selective/adaptive) correct trace guarantee (Def. 7.6 /Def. 7.5) and let n_{bd} be the bound on the number of key queries for an admissible adversary, then our bounded (secret/public tracing)-*EITR *scheme also satisfies the (selective/adaptive) correct trace guarantee (Def. 7.9 /Def. 7.8).*

Proof. Similar to [GKW19] we begin with defining some events of interest. We modify the definition of some events to take into account the constraint that $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$. We drop the subscripts $\mathcal{A}$, ϵ and security parameter λ in the following to keep the notations simple.

- Event Admissible-Adversary: It is defined as the event that the adversary $\mathcal{A}$ makes at most n_{bd} key queries.

- Event Tr (Tracing without correctness): Similar to Corr-Tr, except that we don't need $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$, i.e., Tr is the event that $T^{\mathsf{final}} \neq \phi$ occurs. Denote $\Pr\text{-}\mathsf{Tr} := \Pr[\mathsf{Tr}]$.

- Event Dist-Indx (Position with distinct indices for each key): Defined as the event that there exists $i \in [\lambda]$ such that the i-th index of each key is distinct.

- Event $\mathsf{Dist\text{-}Indx}_i$: It is defined as the event that $i \in [\lambda]$ is the first position such that the i-th index of each key is distinct. By definition, $\mathsf{Dist\text{-}Indx}_i$ are disjoint events for all $i \in [\lambda]$ and $\cup_{i\in[\lambda]}\mathsf{Dist\text{-}Indx}_i = \mathsf{Dist\text{-}Indx}$.
- Event Tr_i (Tracing without correctness in i-th iteration): Let T_i denote the set of (identity, signature) pairs traced in the i-th iteration. The event Tr_i happens if T_i is non-empty.
- Event $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ (Tracing with same signature as that received in key): The event that T_i is not empty and for all $(\mathsf{id}, \sigma) \in T_i$, $(\mathsf{lb}, \mathsf{id})$ was queried for a key and $\mathsf{lb} \notin L$, i.e. $\mathsf{id} \in S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$ and that the key generation oracle output $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},i} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}_i, \mathsf{lb}, (\mathsf{id}, \sigma), j)$ for some j.
- Event $\mathsf{Found\text{-}Good\text{-}r}_i$: This event occurs if flag is set to 1 in the i-th iteration of the tracing algorithm.
- Event $\mathsf{Good\text{-}\tilde{D}}_i$ (Good decoder $\tilde{D}$ during the $\mathsf{Bnd\text{-}Subtrace}$ routine execution in i-th iteration): It is defined as the event that in the i-th iteration, the execution reaches step 3 (that is, it found a 'good' r in the i-th iteration), and the decoder $\tilde{D}$ constructed is an $\epsilon/4$ good decoder for distinguishing messages $m_0 \oplus r$ and $m_1 \oplus r$. Note that if no good r is found in step 3, $\mathsf{Good\text{-}\tilde{D}}_i$ is said to not have happened.

With the above events, the correctness of tracing is argued via the following series of inequalities:

$$\begin{aligned}
\Pr\text{-}\mathsf{Corr\text{-}Tr}(\lambda) \geq & \Pr\text{-}\mathsf{Tr} - \mathsf{negl} && (9.1)\\
\geq & \Pr[\mathsf{Tr} \wedge \mathsf{Dist\text{-}Indx}] - \mathsf{negl} && (9.2)\\
= & \sum_{i\in[\lambda]} \Pr[\mathsf{Tr} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} && (9.3)\\
\geq & \sum_{i\in[\lambda]} \Pr[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} && (9.4)\\
\geq & \sum_{i\in[\lambda]} \Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} && (9.5)\\
\geq & \sum_{i\in[\lambda]} \Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \\
& \qquad \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} && (9.6)\\
\geq & \sum_{i\in[\lambda]} \Pr[\mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary} \\
& \qquad \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} && (9.7)\\
\geq & \sum_{i\in[\lambda]} \Pr[\mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} && (9.8)\\
= & \Pr[\mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}] - \mathsf{negl} && (9.9)\\
\geq & \Pr[\mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary}] - \mathsf{negl} && (9.10)
\end{aligned}$$

Explanation for each of the inequalities is the same as in [GKW19] and is omitted. Here, we argue the transitions from equations (9.3) to (9.4) and (9.4) to (9.5) only, since our definition of event $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ is slightly modified.

- Transition from (9.3) to (9.4) follows from the observation that the event $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ implies the event Tr because by definition, if $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ happens then T_i is non empty and for each (id, σ) pair in T_i, $\mathsf{Sig.Verify}(\mathsf{sig.vk}, \mathsf{id}, \sigma)$ verifies and $\mathsf{id} \in S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$. Hence, the set of traitors T^{final} obtained from T_i will also be non empty.

- Transition from (9.4) to (9.5) is argued via the following claim:

Claim 9.5. *Assume that the underlying* Ind-TR *scheme satisfies correct trace guarantee. Then for all* $i \in [\lambda]$

$$\Pr[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i] \geq \Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl}\,.$$

Proof. We prove the claim by contradiction. We assume that there exists an adversary $\mathcal{A}$ who outputs a good decoder D along with messages m_0, m_1 and a revocation list L such that $\Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \Pr[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i]$ is non-negligible. Then we use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against correct trace guarantee of Ind-TR, defined as follows:

1. At the start of the game, $\mathcal{A}$ outputs $(1^\kappa, 1^{n_{\mathsf{bd}}}, L)$. (In the public trace setting, it outputs L adaptively along with the challenge message).
2. $\mathcal{B}$ sets $n_{\mathsf{index}} = 2{n_{\mathsf{bd}}}^2$, sends $(1^{\kappa+\ell_s}, 1^{n_{\mathsf{index}}}, L)$ to the Ind-TR challenger and gets back Ind.mpk.
3. $\mathcal{B}$ sets $\mathsf{Ind.mpk}_i = \mathsf{Ind.mpk}$ and does the following:
    - Samples $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
    - For $k \in [\lambda] \setminus \{i\}$, sample $(\mathsf{Ind.mpk}_k, \mathsf{Ind.msk}_k) \leftarrow \mathsf{Ind.Setup}(1^{\kappa+\ell_s}, 1^{n_{\mathsf{index}}})$.
    - Set $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_k\}_{k\in[\lambda]})$ and sends mpk to $\mathcal{A}$.
4. $\mathcal{A}$ issues at most n_{bd} key queries. For each key query $(\mathsf{lb}, \mathsf{id})$ by $\mathcal{A}$, $\mathcal{B}$ does the following:
    - Computes $\mathsf{Sig.Sign}(\mathsf{sig.sk}, \mathsf{id})$ and sets $\mathsf{id}' = (\mathsf{id}, \sigma)$.
    - For $k \in [\lambda]$, randomly chooses $j_k \leftarrow [n_{\mathsf{index}}]$ and sends a key query $(\mathsf{lb}, \mathsf{id}', j_i)$ to the Ind-TR challenger and gets back $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j_i}$. $\mathcal{B}$ sets $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},i} = \mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j_i}$ and for $k \in [\lambda] \setminus \{i\}$, it computes $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},k} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}_k, \mathsf{lb}, \mathsf{id}', j_k)$.
    - Sends $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = (\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},1}, \ldots, \mathsf{sk}_{\mathsf{lb},\mathsf{id},\lambda})$ to $\mathcal{A}$.
5. If i is not the first position for which the i-th position indices (j_i's) are distinct for all the key queries, then $\mathcal{B}$ outputs a random decoder and quits the game.
6. In the end, $\mathcal{A}$ outputs a decoder D along with messages m_0, m_1.
7. $\mathcal{B}$ runs $\mathsf{Bnd\text{-}isGoodDecoder}(\{\mathsf{Ind.mpk}\}_{j\in[\lambda]}, 1^y, m_0, m_1, L, r)$ for uniformly and independently sampled r until it finds a r for which Bnd-isGoodDecoder algorithm outputs 1. If Bnd-isGoodDecoder algorithm does not output 1 even after $\lambda \cdot y$ attempts, then $\mathcal{B}$ outputs a random decoder and quits.
8. $\mathcal{B}$ constructs decoder $\tilde{D}$ as defined in Figure 11 and sets $\tilde{m}_b = m_b \oplus r$ for $b \in \{0, 1\}$ and sends $(\tilde{D}, \tilde{m}_0, \tilde{m}_1, L)$ to the Ind-TR challenger.

Now let us analyze the probability that $\mathcal{B}$ outputs a $1/4y$ good decoder box. This happens if (i) i is the first position such that the i-th position indices are different for all the key queries (ii) Bnd-isGoodDecoder outputs 1 for some r and $\tilde{D}$ is $\epsilon/4 = 1/4y$ good decoder for distinguishing $m_0 \oplus r$, $m_1 \oplus r$. First event is same as $\mathsf{Dist\text{-}Indx}_i$ and second event is same as $\mathsf{Good\text{-}\tilde{D}}_i$. Hence, the probability that $\mathcal{B}$ outputs a $1/4y$ good decoder box is $\Pr[\mathsf{Good}\text{-}\tilde{\mathsf{D}}_i \wedge \mathsf{Dist}\text{-}\mathsf{Indx}_i]$. Then, by correct trace guarantee of Ind-TR,

$$\begin{aligned}\Pr[\mathsf{Corr}\text{-}\mathsf{Tr}\text{-}\mathsf{Sig}_i \wedge \mathsf{Dist}\text{-}\mathsf{Indx}_i] &\geq \Pr[\mathsf{Good}\text{-}\tilde{\mathsf{D}}_i \wedge \mathsf{Dist}\text{-}\mathsf{Indx}_i] - \mathsf{negl}\\ &\geq \Pr[\mathsf{Good}\text{-}\tilde{\mathsf{D}}_i \wedge \mathsf{Admissible}\text{-}\mathsf{Adversary} \wedge \mathsf{Dist}\text{-}\mathsf{Indx}_i] - \mathsf{negl}\,.\end{aligned}$$

This contradicts our assumption that $\Pr[\mathsf{Good}\text{-}\tilde{\mathsf{D}}_i \wedge \mathsf{Admissible}\text{-}\mathsf{Adversary} \wedge \mathsf{Dist}\text{-}\mathsf{Indx}_i] - \Pr[\mathsf{Corr}\text{-}\mathsf{Tr}\text{-}\mathsf{Sig}_i \wedge \mathsf{Dist}\text{-}\mathsf{Indx}_i]$ is non-negligible, hence the proof. □

□

10 Unbounded Trace and Revoke with Embedded Identities

In this section we show how to construct unbounded (secret/public tracing)-EITR scheme from a bounded (secret/public tracing)-EITR scheme. The transformation technique and the security analysis in this section is adapted from [GKW19] with modifications to incorporate the revocation list. We present the construction and proofs for *secret-key* trace setting primarily and also outline the differences in the public-key trace setting simultaneously.

10.1 Construction

Let BD-TR = (BD.Setup, BD.KeyGen, BD.Enc, BD.Dec, BD.Trace) be a bounded (secret/public tracing)-EITR scheme for identity space $\mathcal{ID} = \{0,1\}^\kappa$. We construct an unbounded (secret/public tracing)-EITR scheme as follows.

$\mathsf{Setup}(1^\lambda, 1^\kappa) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following:

1. For $j = 1$ to λ, sample $(\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, n_{\mathsf{bd}} = 2^j)$.
2. Output $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$ and $\mathsf{msk} = \{\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j\}_{j\in[\lambda]}$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{lb},\mathsf{id}}$. The KeyGen algorithm does the following:

1. Parse $\mathsf{msk} = \{\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j\}_{j\in[\lambda]}$.
2. For $j = 1$ to λ, it computes $\mathsf{BD.sk}_j \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}, \mathsf{id})$.
3. Returns $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{BD.sk}_j\}_{j\in[\lambda]}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. The encryption algorithm does the following:

1. Parse $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$.
2. Secret share m in λ shares as follows. For $j = 1$ to $\lambda - 1$, randomly sample $r_j \leftarrow \mathcal{M}$ and set $r_\lambda = m \oplus r_1 \oplus \ldots \oplus r_{\lambda-1}$.
3. For $j = 1$ to λ, compute $\mathsf{BD.ct}_j = \mathsf{BD.Enc}(\mathsf{BD.mpk}_j, r_j, L)$.
4. Output $\mathsf{ct} = \{\mathsf{BD.ct}_j\}_{j\in[\lambda]}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{id}}, \mathsf{ct}, L) \to m'$. The decryption algorithm does the following:

1. Parse $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{BD.sk}_j\}_{j\in[\lambda]}$ and $\mathsf{ct} = \{\mathsf{BD.ct}_j\}_{j\in[\lambda]}$.

Algorithm $\mathsf{isGoodDecoder}^D(\mathsf{key}, 1^y, m_0, m_1, r, L, i)$
Inputs: keys $\mathsf{key} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and a position-index $i \in [\lambda]$.
Output : $0/1$.

1. Set $\mathsf{count} = 0$. Let $\epsilon = 1/y$.
2. For $j = 1$ to $\lambda \cdot y$:
   - Sample $\lambda - 1$ messages, $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_\lambda$ randomly such that $\oplus_{k\in[\lambda]\setminus\{i\}} r_k = r$.
   - Sample $b \leftarrow \{0,1\}$, and compute ciphertexts $\mathsf{BD.ct}_k = \mathsf{BD.Enc}(\mathsf{BD.mpk}_k, r_k, L)$ for $k \in [\lambda] \setminus \{i\}$ and $\mathsf{BD.ct}_i = \mathsf{BD.Enc}(\mathsf{BD.mpk}_i, r \oplus m_b, L)$.
   - Query D on $\mathsf{ct} = (\mathsf{BD.ct}_1, \ldots, \mathsf{BD.ct}_\lambda)$. Let b' be the response of D.
   - If $b' = b$, set $\mathsf{count} = \mathsf{count} + 1$.
3. If $\mathsf{count}/(\lambda \cdot y) \geq 1/2 + \epsilon/3$, then output 1, else output 0.

Figure 12: Algorithm isGoodDecoder for Unbounded EITR

2. For $j = 1$ to λ, compute $r'_j = \mathsf{BD.Dec}(\mathsf{BD.sk}_j, \mathsf{BD.ct}_j, L)$.
3. If any of the decryption fails then output $m' = \perp$, else output $m' = \bigoplus_{j\in[\lambda]} r'_j$.

$\mathsf{Trace}^D(\mathsf{tk}, y, \mathsf{Q_{bd}}, m_0, m_1, L) \rightarrow T$. The trace algorithm uses two algorithms isGoodDecoder and SubTrace defined in Figures 12 and 13, respectively as subroutines. The tracing algorithm is as follows.

1. Parse tk as $\mathsf{msk} = \{\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j\}_{j\in[\lambda]}$ and let $\mathsf{BD.tk}_j = \mathsf{BD.msk}_j$ for $j \in [\lambda]$. (For the public trace setting $\mathsf{tk} = \mathsf{mpk}$ and $\mathsf{BD.tk}_j = \mathsf{BD.mpk}_j$)
2. Set $j = \lceil \log \mathsf{Q_{bd}} \rceil$.
3. Set $\mathsf{flag} = 0$. For $\mathsf{itr} = 1$ to $\lambda \cdot y$, do the following
   (a) Choose a random message $r \leftarrow \mathcal{M}$.
   (b) Run isGoodDecoder as
   $\mathsf{flag} \leftarrow \mathsf{isGoodDecoder}^D(\{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$
   (c) If $\mathsf{flag} = 1$, break. Else, continue.
4. If $\mathsf{flag} = 1$, run SubTrace as $T \leftarrow \mathsf{SubTrace}^D(\{\mathsf{BD.mpk}_j, \mathsf{BD.tk}_j\}_{j\in[\lambda]}, 1^y,$ $m_0, m_1, r, L, j)$. Else, set $T = \phi$.
5. Output T.

Correctness. We show that the above construction of bounded (secret/public tracing)-EITR satisfies correctness (Def. 7.1) via the following theorem.

Theorem 10.1. *Assume* BD-TR *is a correct bounded (secret/public tracing)-*EITR *scheme then the above construction of unbounded (secret/public tracing)-*EITR *scheme is correct.*

Algorithm $\mathsf{SubTrace}^{D}(\mathsf{key}, 1^y, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{BD.mpk}_j, \mathsf{BD.tk}_j\}_{j \in [\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and a position-index $i \in [\lambda]$.

Output : $T \subseteq \{0,1\}^{\kappa}$.

1. Define oracle $\tilde{D}[\{\mathsf{BD.mpk}_j\}_{j \in [\lambda]}, r, L, i]$ as in Figure 14.
2. Output $T \leftarrow \mathsf{BD.Trace}^{\tilde{D}}(\mathsf{BD.tk}_i, 4y, m_0 \oplus r, m_1 \oplus r, L)$.

Figure 13: Algorithm: SubTrace for Unbounded EITR

Algorithm $\tilde{D}^{D}[\mathsf{key}, r, L, i]$

Hardwired values: keys $\mathsf{key} = \{\mathsf{BD.mpk}_j\}_{j \in [\lambda]}$, message r, revocation list L and a position-index $i \in [\lambda]$.
Inputs: BD.ct.
Output : 0/1
On input BD.ct, the $\tilde{D}$ oracle does the following:

- It first shares r in $\lambda - 1$ components as follows: it chooses $\lambda - 1$ random messages r_k for $k \in [\lambda] \setminus \{i\}$, such that $\oplus_{k \in [\lambda] \setminus \{i\}} r_k = r$.
- It computes $\mathsf{BD.ct}_k = \mathsf{BD.Enc}(\mathsf{BD.mpk}_k, r_k, L)$ for $k \in [\lambda] \setminus \{i\}$.
- Sets $\mathsf{ct} = (\mathsf{BD.ct}_1, \ldots, \mathsf{BD.ct}_{i-1}, \mathsf{BD.ct}, \mathsf{BD.ct}_{i+1}, \ldots, \mathsf{BD.ct}_{\lambda})$.
- Outputs $b' \leftarrow D(\mathsf{ct})$.

Figure 14: Oracle $\tilde{D}$ for Unbounded EITR

Proof. For all $k \in [\lambda]$, if $\mathsf{BD.ct}_k \leftarrow \mathsf{BDEnc}(\mathsf{BD.mpk}_k, r_k, L)$ and $\mathsf{BD.sk}_k \leftarrow \mathsf{BDKeyGen}(\mathsf{BD.msk}_k, (\mathsf{lb}, \mathsf{id}))$, we have $r_k \leftarrow \mathsf{BD.Dec}(\mathsf{BD.sk}_k, \mathsf{BD.ct}_k, L)$, as long as $\mathsf{id} \notin L$, from the correctness of the underlying BD-TR scheme. Hence, the decryption of $\mathsf{ct} = (\mathsf{BD.ct}_1, \ldots, \mathsf{BD.ct}_\lambda)$ correctly outputs m as $r_1 \oplus \cdots \oplus r_\lambda$. □

Efficiency. We can instantiate the above construction by the bounded public/secret tracing-EITR scheme in Sec. 9. Since the above construction is simple λ times repetition of the underlying bounded EITR scheme, the parameter size of the scheme is as follows.

Secret Tracing Setting. In the secret tracing setting, we have

$$|\mathsf{mpk}|, |\mathsf{ct}|, |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|).$$

Public Tracing Setting. In the public tracing setting, we have

$$|\mathsf{mpk}|, |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathsf{lb}|),\ |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|).$$

10.2 Security

In this section, we prove that our construction of unbounded (secret/public tracing)-EITR scheme is secure.

IND-CPA security.

Theorem 10.2. *If the underlying* BD-TR *scheme is (selective/adaptive)* IND-CPA *secure bounded (secret/public tracing)-*EITR *scheme, then the above construction of unbounded (secret/public tracing)-*EITR *is (selective/adaptive)* IND-CPA *secure.*

Proof. We show that if there exists a PPT adversary that breaks the IND-CPA security of unbounded EITR scheme, then we can use $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that breaks IND-CPA security of the underlying BD-TR scheme. □

The reduction is defined as follows:

1. $\mathcal{B}$ first gets $1^\kappa, L$ from the adversary $\mathcal{A}$ (in the public trace setting, $\mathcal{A}$ outputs L adaptively along with the challenge messages). It then sets $n_{\mathsf{bd}} = 2$ and sends $(1^\kappa, 1^{n_{\mathsf{bd}}}, L)$ to the BD-TR challenger. The BD-TR challenger returns BD.mpk.

2. $\mathcal{B}$ sets $\mathsf{BD.mpk}_1 = \mathsf{BD.mpk}$, generates $((\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, 2^j))$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j \in [\lambda]}$ to $\mathcal{A}$.

3. When $\mathcal{A}$ makes a key query $(\mathsf{lb}, \mathsf{id})$, $\mathcal{B}$ sends $(\mathsf{lb}, \mathsf{id})$ to the BD-TR challenger as a key query and sets the secret key it receives as $\mathsf{BD.sk}_1$. $\mathcal{B}$ then computes $\mathsf{BD.sk}_j \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}, \mathsf{id})$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{BD.sk}_j\}_{j \in \lambda}$ to $\mathcal{A}$.

4. When $\mathcal{A}$ sends the challenge query (m_0, m_1), $\mathcal{B}$ samples $r_j \leftarrow \mathcal{M}$ for $j \in \{2, \cdots, \lambda\}$, sets $m'_b = \bigoplus_{j>1} r_j \oplus m_b$ for $b \in \{0, 1\}$ and sends (m'_0, m'_1) as challenge query to the BD challenger and and sets the returned ciphertext as $\mathsf{BD.ct}_1$. $\mathcal{B}$ then computes $\mathsf{BD.ct}_j \leftarrow \mathsf{BD.Enc}(\mathsf{BD.mpk}_j, r_j, L)$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{ct} = \{\mathsf{BD.ct}_j\}_{j \in [\lambda]}$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ sends b' to the BD-TR challenger.

We observe that $\mathcal{B}$ issues a key query $(\mathsf{lb}, \mathsf{id})$ to the BD-TR challenger only when $\mathcal{A}$ makes a key query on $(\mathsf{lb}, \mathsf{id})$. So , by the admissibility of $\mathcal{A}$, we have $\mathsf{lb} \in L$ and thus $\mathcal{B}$ is admissible in the IND-CPA game of BD-TR. Also, if $\mathcal{A}$ has an advantage ϵ in distinguishing the encryptions of m_0, m_1, then clearly $\mathcal{B}$ has the same advantage in distinguishing the encryptions of m'_0, m'_1 and thus breaking the IND-CPA security of BD-TR.

Secure Tracing Analysis. Now we show that our construction satisfies false trace and correct trace guarantees.

False Trace Guarantee.

Theorem 10.3. *Assume that the underlying (secret/public tracing)-*BD-TR *scheme satisfies (selective/adaptive) false trace guarantee, then our construction of unbounded (secret/public tracing)-*EITR *satisfies (selective/adaptive) false trace guarantee as defined in Def. 7.12/Def. 7.11.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$, polynomial $p(\lambda)$ and non-negligible functions $\epsilon(\cdot), \delta(\cdot)$ such that $\Pr[\mathsf{Fal\text{-}Tr}]_{\mathcal{A},\epsilon,p} \geq \delta(\lambda)$, then we can build a PPT reduction $\mathcal{B}$ that can break the false trace guarantee of BD-TR. The reduction is as follows:

1. $\mathcal{B}$ first receives $1^\kappa, L$ from the adversary $\mathcal{A}$. (In the public trace setting, $\mathcal{A}$ outputs L adaptively along with the challenge messages). It sets $i = \lceil \log p(\lambda) \rceil$, sends $(1^\kappa, 1^{2^i}, L)$ to the BD-TR challenger and sets the public key it receives as $\mathsf{BD.mpk}_i$. $\mathcal{B}$ generates $(\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, 2^j)$ for $j \in [\lambda] \setminus \{i\}$ and sends $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes a key query $(\mathsf{lb}, \mathsf{id})$, $\mathcal{B}$ forwards it as a key query to the BD-TR challenger and sets the secret key it receives as $\mathsf{BD.sk}_i$. $\mathcal{B}$ generates $\mathsf{BD.sk}_j \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}, \mathsf{id})$ for $j \in [\lambda] \setminus \{i\}$ and sends $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{BD.sk}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

3. Finally, when $\mathcal{A}$ outputs a decoder box D and messages m_0, m_1 , $\mathcal{B}$ runs $\mathsf{isGoodDecoder}((\mathsf{BD.mpk}_j)_{j\in[\lambda]}, 1^{1/\epsilon}, m_0, m_1, r, L, i)$ as defined in Figure 12, for $\lambda \cdot y$ many choices of r, until it finds an r s.t isGoodDecoder outputs 1.

4. $\mathcal{B}$ constructs a decoding box $\tilde{D}$ as defined in Figure 13, sends $(\tilde{D}, m_0 \oplus r, m_1 \oplus r)$ to the BD-TR challenger.

We observe that $\tilde{D}$ uses the decoder D as a subroutine and it returns the response of D as its output. So, if $\mathcal{A}$ outputs (D, m_0, m_1, L) such that the false trace guarantee does not hold with non-negligible probability, then $\mathcal{B}$ breaks the false trace guarantee of the underlying BD-TR scheme. □

Correct Trace Guarantee.

Theorem 10.4. *Assume that the underlying (secret/public tracing)-*BD-TR *scheme satisfies (selective/adaptive) correct trace guarantee, then our construction of unbounded (secret/public tracing)-*EITR *satisfies (selective/adaptive) correct trace guarantee as defined in Def. 7.12/Def. 7.11.*

Proof. Let $i = \lceil \log p(\lambda) \rceil$ and let $S, S_{\mathcal{ID}}, T_{\mathsf{lb}}$ be as defined in Def. 7.11. Consider the following events

Event $\mathsf{Cor\text{-}Tr}_i$: is defined as the event that the SubTrace algorithm, when run for position i outputs a correct traitor set T, i.e. $|T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$.

Event $\mathsf{Good\text{-}\tilde{D}}_i$: is defined as the event that the flag is set to 1 in step 3 and the decoder $\tilde{D}$ defined in Fig 14 is $\epsilon/4$ good decoder.

Event $\mathsf{Found\text{-}Good\text{-}r}_i$: is defined as the event that the isGoodDecoder algorithm, when run for position i outputs 1.

With these definitions, the correctness is argued via following series of inequalities:

$$\begin{aligned}
\Pr\text{-}\mathsf{Corr\text{-}Tr}(\lambda) &= \Pr[\mathsf{Corr\text{-}Tr}_i] && (10.1)\\
&\geq \Pr[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i] && (10.2)\\
&\geq \Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (10.3)\\
&\geq \Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (10.4)\\
&\geq \Pr[\mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (10.5)\\
&\geq \Pr[\mathsf{Good\text{-}Decoder} \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (10.6)
\end{aligned}$$

Equation (10.1) and (10.2) follow directly from the definition of the events involved. Equation (10.3) follows from the following claim:

Claim 10.4.1. *If* BD-TR *guarantees correct tracing then*

$$\Pr\left[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|\right] - \Pr[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i] \leq \mathsf{negl}(\lambda).$$

Proof. We show that if there exists an adversary $\mathcal{A}$ who outputs a good decoder along with messages m_0, m_1 and a revocation list L such that $\Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|] - \Pr[\mathsf{Cor\text{-}Tr}_i]$ is non negligible then we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against correct trace guarantee of the underlying BD-TR. The reduction is defined as follows:

1. $\mathcal{B}$ first gets $(1^\kappa, L)$ from $\mathcal{A}$ (in the public trace setting, $\mathcal{A}$ outputs L adaptively along with the challenge messages). It sends $(1^\kappa, 1^{2^i}, L)$ to the BD-TR challenger and sets the public key it receives as $\mathsf{BD.mpk}_i$.

2. For $j \in [\lambda] \setminus \{i\}$, $\mathcal{B}$ generates $(\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\kappa, 2^j)$; sets $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$, and sends mpk to $\mathcal{A}$.

3. For each key query $(\mathsf{lb}, \mathsf{id})$ from $\mathcal{A}$, $\mathcal{B}$ sends $(\mathsf{lb}, \mathsf{id})$ as a key query to the BD-TR challenger and sets the key received as $\mathsf{BD.sk}_i$. For $j \in [\lambda] \setminus \{i\}$, it computes $\mathsf{BD.sk}_j$ itself using $\mathsf{BD.msk}_j$. It sends $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{BD.sk}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a decoder D along with messages m_0, m_1.

5. $\mathcal{B}$ does the following:

   - If the number of key queries by $\mathcal{A}$ exceeds $p(\lambda)$, then $\mathcal{B}$ outputs a random decoder and quits.
   - Else, $\mathcal{B}$ runs $\mathsf{isGoodDecoder}(\{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, i)$ for uniformly and independently sampled r until it finds a r for which isGoodDecoder algorithm outputs 1. If isGoodDecoder algorithm does not output 1 even after $\lambda \cdot y$ attempts, then $\mathcal{B}$ outputs a random decoder and quits.

- Else, $\mathcal{B}$ constructs decoder $\tilde{D}$ as defined in Figure 14 and sets $\tilde{m}_b = m_b \oplus r$ for $b \in \{0,1\}$.

6. $\mathcal{B}$ sends $(\tilde{D}, \tilde{m}_0, \tilde{m}_1)$ to the BD-TR challenger.

Now we analyze the probability that $\mathcal{B}$ outputs a good decoder. Observe that $\mathcal{B}$ does not abort and outputs a genuine decoder when both $\mathsf{Found\text{-}Good\text{-}r}_i$ and $|S_{ID}| \leq p(\lambda)$ happens. Since the decoder returned by $\mathcal{B}$ is the same decoder as defined in the SubTrace algorithm, we get that the probability that $\mathcal{B}$ outputs a good decoder is $\Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|]$. Furthermore, the probability that the BD.Trace algorithm outputs correct set of traitors T (using the decoder returned by $\mathcal{B}$) is same as $\Pr[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i]$. Hence, if $\Pr[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|] - \Pr[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i]$ is non negligible, then it breaks the security of correct trace guarantee of the underlying BD-TR scheme. □

Equation (10.4) again follows directly from the definition. Argument for transition from equation (10.4) to (10.5) and (10.5) to (10.6) is same as that of transition from (9.8) to (9.9) and (9.9) to (9.10), respectively. This completes the proof. □

11 Extension to Super-Polynomial Size Revocation List

While our main focus in the paper is on the case where the revocation list L is of polynomial size, we can consider an extension where it is of super-polynomial size. In particular, we consider the setting where L has efficient representation by a circuit C_L of polynomial size. Namely, we have $C_L(\mathsf{lb}) = 0$ for revoked label lb and $C_L(\mathsf{lb}) = 1$ for non-revoked label lb. We assume that the depth of C_L is bounded by some polynomial $\bar{d}$. Namely, $C_L \in \mathcal{C}_{|\mathsf{lb}|,\bar{d}}$. This assumption is necessary because we will use kpABE (resp., cpABE) with the same restriction to generate a secret key (resp., a ciphertext) associated with C_L.

EITR scheme for this setting can be obtained both in the secret and the public tracing settings very similarly to the case where the revocation list is of polynomial size. Namely, we first construct (secret key/public key) RPE with super-polynomial revocation list and then apply the chain of conversions (Sec. 8, 9, and 10). The conversions work almost without change with the natural adaptation of replacing L with C_L and the membership check $\mathsf{lb} \overset{?}{\in} L$ with $C(\mathsf{lb}) \overset{?}{\in} 0$. The constructions of (pubic key/secret key) RPE are also almost the same as those in (Sec. 4/Sec. 5 and 6) with the natural adaptation of generating kpABE secret key for C_L[27] in Sec. 4 and 5 and replacing the condition $\mathsf{lb} \notin L$ in Eq. (6.1) defining $C_{L,\mathsf{RMFE.ct}}$ with $C_L(\mathsf{lb}) = 1$. The main difference is that we have to assume sub-eponential LWE assumption instead of (polynomial) LWE assumption for both secret key and public key settings here, because we need adaptive security for the underlying kpABE. We give further details in the following.

- In the public key setting, indistinguishability of $\mathsf{Hybrid}_{6,a}$ and Hybrid_7 shown in Claim 4.2.7, where post-challenge key queries are dealt with, cannot be proven any more if we only assume selective security for kpABE. The reason why the original proof does not go through is that we have to deal with the kpABE queries in the order of key first and ciphertext later. With polynomial size L, this does not pose a problem because when the adversary chooses L, all the labels for which we use kpABE security are in L and we can perform a hybrid argument over these labels. However, this is not possible for super-polynomial size L. To deal with the queries in this order, we assume adaptive

[27] Originally, we start from the revocation list L and then construct the circuit C_L that hardwires L into it. Here, we directly use the circuit C_L that efficiently represents the super-polynomial set L.

security (as per Definition 2.16) for kpABE. Then, the indistinguishability of the games can easily be shown by changing the post-challenge kpABE ciphertexts associated with lb with $C_L(\mathsf{lb}) = 0$ one by one.

- In the secret key setting, we also encounter similar difficulty. In particular, indistinguishability of Hybrid_1 and Hybrid_2 shown in Claim 5.8.2 cannot be proven any more if we only assume selective security for kpABE by exactly the same reason. We can overcome the problem by assuming adaptive security for kpABE.

Finally, we briefly discuss the parameter size of the resulting EITR scheme. The parameter size of the resulting scheme is the same as the case of polynomial size revocation list except that they rely on $\bar{d}$. Notably, they are independent from the size of the circuits being supported inheriting the succinctness properties of the underlying kpABE and cpABE.

Acknowledgements. This work was partly supported by the DST "Swarnajayanti" fellowship, Cybersecurity Center of Excellence, IIT Madras, National Blockchain Project, European Union Horizon 2020 Research and Innovation Program Grant 780701, BPIFrance in the context of the national project RISQ (P141580), and the ANR AMIRAL project (ANR-21-ASTR-0016). The fourth author is partially supported by JST AIP Acceleration Research JPMJCR22U5 and JSPS KAKENHI Grant Number 19H01109, Japan.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO*, 2015.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *Eurocrypt*, 2019.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. IACR Cryptology ePrint Archive, 2015:730, 2015.

[ALP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *PKC*. Springer, 2011.

[AM18] Shweta Agrawal and Monosij Maitra. Fe and io for turing machines from minimal assumptions. In *TCC*. Springer, 2018.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *TCC*, 2019.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from lwe and pairings in the standard model. In *TCC*, 2020.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and lwe. In *EUROCRYPT*, 2020.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, 2014.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, 2005.

[BKS16] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *EUROCRYPT*. Springer, 2016.

[BS18] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31(1):202–225, 2018.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *FOCS*, 2015.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, 2006.

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, 2014.

[BZ17] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, 2017.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, 1994.

[CIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO*, 2013.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, 2013.

[CVW+18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *TCC*, 2018.

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct lwe sampling, random polynomials, and obfuscation. In *TCC*, 2021.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO*, 1993.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In *TCC*, 2019.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *STOC*, 2021.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *CRYPTO*, 2016.

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In *EUROCRYPT*, 2017.

[GQWW19] Rishab Goyal, Willy Quach, Brent Waters, and Daniel Wichs. Broadcast and trace with n^{ϵ} ciphertext size from standard assumptions. In *Crypto*, 2019. https://eprint.iacr.org/2019/636.

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In *TCC*, 2016.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013.

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In *PKC*, 2019.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over large fields, dlin, and constant depth prgs. In *EUROCRYPT*, 2022.

[KS20] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. *Journal of Cryptology*, 33(2):406–458, 2020.

[KW20] Sam Kim and David J Wu. Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In *ASIACRYPT*, 2020.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In *TCC*, 2016.

[NP10] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. *International Journal of Information Security*, 9(6):411–424, 2010.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: how to embed arbitrary information in a key. In *EUROCRYPT*, 2016.

[Tak14] Katsuyuki Takashima. Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In *SCN*, 2014.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In *CRYPTO*, 2019.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In *Crypto*, 2022.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and cp-abe from evasive lattice assumptions. In *Eurocrypt*, 2022.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *EUROCRYPT*. Springer, 2021.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, 2017.

[Zha20] Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In *CRYPTO*, 2020.