Public-Key Puncturable Encryption: Modular and Compact Constructions

Shi-Feng Sun[1,2], Amin Sakzad[1], Ron Steinfeld[1], Joseph Liu[1], and Dawu Gu[3]

[1] Faculty of Information Technology,
Monash University, Clayton, Australia
[2] Data61, CSIRO, Melbourne, Australia
[3] Department of Computer Science & Engineering,
Shanghai Jiao Tong University, Shanghai, China
{shifeng.sun, amin.sakzad, ron.steinfeld, joseph.liu}@monash.edu
dwgu@sjtu.edu.cn

Abstract. We revisit the method of designing public-key puncturable encryption schemes and present a generic conversion by leveraging the techniques of distributed key-distribution and revocable encryption. In particular, we first introduce a refined version of identity-based revocable encryption, named *key-homomorphic identity-based revocable key encapsulation mechanism with extended correctness*. Then, we propose a generic construction of *puncturable key encapsulation mechanism* from the former by merging the idea of distributed key-distribution. Compared to the state-of-the-art, our generic construction supports unbounded number of punctures and multiple tags per message, thus achieving more *fine-grained* revocation of decryption capability. Further, it does not rely on *random oracles*, not suffer from *non-negligible* correctness error, and results in a variety of efficient schemes with distinct features. More precisely, we obtain the *first* scheme with very *compact* ciphertexts in the standard model, and the *first* scheme with support for both *unbounded* size of tags per ciphertext and *unbounded* punctures as well as *constant-time* puncture operation. Moreover, we get a comparable scheme proven secure under the standard DBDH assumption, which enjoys both faster encryption and decryption than previous works based on the same assumption, especially when the number of tags associated with the ciphertext is large.
Keywords: Functional Encryption; Puncturable Encryption; Forward Security

1 Introduction

Public Key Encryption (PKE) is a critical cryptographic tool for protecting the confidentiality of messages transmitted over insecure communication channels, which has been widely employed in practice such as messaging services. It is commonly agreed that the standard security for PKE is indistinguishability against chosen-ciphertext attack (IND-CCA) that is guaranteed under the perfect secrecy of secret keys. However, as more and more cryptographic applications are

performed on poorly protected mobile devices, the threat of key compromise to attackers through virus or physical access becomes more and more acute nowadays, and thus will lead to the lost of the security guarantees.

To deal with such kind of threat, numerous methods have been introduced, including key-insulated cryptography [22], threshold cryptography [21], proactive cryptography [36] and forward security [27,11]. As a promising approach, forward security has been considered in a variety of cryptographic primitives, since the initial introduction in the context of key exchange protocol [27] in 1989. However, the first forward secure PKE (FS-PKE) was proposed by Canetti et al. [11] in 2003. In general, a forward secure PKE scheme is usually equipped with an efficient update algorithm, by which the current secret key can be altered so that it cannot be used to recover *past* messages. In other words, the decryption capability for previous ciphertexts is revoked by updating the secret key.

Motivated by the problem that existing forward secure PKE schemes cannot support fine-grained revocation of decryption capability (e.g., removing decryption capability for any individual ciphertext or all ciphertexts sent during a special period), Green and Miers [26] introduced a new form of PKE — Public-key Puncturable Encryption (PPE) — for achieving forward secure asynchronous messaging. In general, this primitive supports multiple tags per message (or ciphertext), which may contain a unique message identifier (e.g., GUID) and some additional metadata (e.g., the sender identity). This feature endows the recipient with the ability of not only revoking individual ciphertext but also the entire classes of ciphertexts (e.g., all ciphertext from the same sender), so it can achieve forward security at a fine-grained level.

Briefly, PPE can be seen as a form of tag-based encryption [31] added with an efficient key-update algorithm called Puncture algorithm. In particular, this algorithm takes as input the current secret key SK and a tag t and outputs a new (punctured) secret key SK' that can decrypt all ciphertexts except for those encrypted under tag t. By this procedure, the secret key can be punctured repeatedly and sequentially on many distinct tags, thus revoking the decryption capability for the ciphertexts encrypted under (any of) these tags. Based on the Key-Policy Attribute-Based Encryption (KP-ABE) scheme [35], Green and Miers proposed the first concrete PPE scheme [1] in the random oracle model. Further to reduce the decryption cost of PPE scheme alone, they put forward a new variant of FS-PKE scheme, named Puncturable Forward Secure PKE (PFSE), by combining their PPE scheme with a variant of Canetti et al. FS-PKE scheme [11]. Subsequently, Günther et al. in [28] introduced the key encapsulation version of PFSE (PFSKEM) and proposed a generic constriction of PFSKEM from any one-time signature and hierarchical identity-based key encapsulation (HIBKEM) scheme [8] with special properties. In this work, we are more interested in PPE itself. Recently, it has been employed widely to achieve other cryptographic goals, such as constructing forward secure 0-RTT protocols [19], backward private searchable encryption [10], forward secure proxy re-encryption [20], and

[1] The proposed PPE scheme supports an arbitrary number of punctures, and the decryption cost is linear in the number of punctures.

public-key watermarking schemes [15]. This demonstrates that PPE is a useful and valuable cryptographic tool.

However, the existing PPE schemes suffer from different shortcomings. In more details, the instantiations from [12,15] are given on the basis of indistinguishability obfuscation, which are more feasibility results than practical solutions. The state-of-the-art construction is from Derler et al. [19,18], in which they introduced a relaxed variant of PPE termed Bloom Filter Encryption (BFE). Specifically, their basic construction from Identity-Based Encryption (IBE) [9] features both efficient puncture and decryption procedure, but has a large ciphertext expansion. Moreover, they presented two *generic* constructions from Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [7] and Identity-Based Broadcast Encryption (IBBE) [17] under the same framework, thus achieving different tradeoffs between (public) key size and ciphertext size. For example, the design from IBBE allows us to obtain the BFE schemes with compact ciphertexts. Due to relying heavily on Bloom filter [33], however, BFE is subject to *non-negligible* correctness error. As argued in [15], this does not affect the application of BFE to designing efficient forward-secure 0-RTT protocols, but may limit its deployment in the scenarios requiring *negligible* correctness error. In addition, BFE schemes only support a *pre-determined* number of punctures, due to the inherent properties of Bloom filters, and a *unique* tag per ciphertext. This makes it less fine-grained than the PPE scheme in [26], as for example it cannot support the revocation of *all* ciphertexts from a single sender. In contrast, the scheme by Green and Miers avoids these drawbacks, but still suffers from some others, such as (1) the number of tags per ciphertext is bounded by some pre-determined parameter $d \in \mathbb{N}$ at the setup, (2) the size of both public key and ciphertext are linear in the pre-determined integer d, and (3) the security is achieved in the random oracle model.

As great effort has been made to improve the performance, security and/or functionality of attribute-based encryption (e.g., [4,34,38,2,14,1,24]) during the past decade, it is also significant to design puncturable encryption with nice features like unbounded tags (or attributes) per message. As emphasized in [34], it is highly desirable in practice to make the parameters for secret key and encryption unbounded by the public parameters fixed at setup, otherwise the public parameter size should be very huge and the scheme will be less flexible. This feature is also important for PPE applications. For example, in asynchronous messaging the decryption capability of metadata (possibly containing a huge number of attributes) encrypted with such scheme can be flexibly revoked by puncturing secret key on any type of attribute.

Based on previous discussions and the systemized work of [19], the natural questions include:

1. How to design efficient PPE schemes with as many desired features as possible (e.g., negligible correctness error, unbounded punctures and compact ciphertext)?
2. Is it possible to generically construct PPE with negligible correctness errors from other cryptographic primitives?

In this work, we make affirmative progress to above questions by leveraging the idea of distributed key-distribution and the revocation encryption technique. In particular, we propose a generic construction of puncturable key encapsulation mechanism by embedding the distributed key-distribution technique to a key encapsulation version of revocation encryption system, and thus obtain a variety of concrete PPE schemes featuring distinct characteristics. The high-level idea is described below, and the main contributions are summarized in Section 1.2.

1.1 Technical Overview

In a PPE scheme, the secret key is punctured gradually; the punctured secret key is updated as a new tag (to be punctured) arrives. The functionality of PPE requires that the ciphertext can be decrypted only if no tag attached to the ciphertext has been punctured. Our idea is inspired by the design of symmetric puncturable encryption [37] and the distributed symmetric key-distribution [32,23,16], so we concentrate on the key encapsulation version of PPE in this work. To support unbounded punctures, the intuition is to distribute an (encapsulated) symmetric key in a similar way as in [26]. Basically, the idea is to produce a share of the encryption/encapsulated key on-the-fly and to reconstruct this key from all shares for completing the decryption. Similar to [37], one share corresponds to one master secret key and each master key is used to puncture a unique tag. In this framework, the crucial point is to make sure that the share (indirectly) associated with tag t cannot be recovered once t belongs to the tag list T of the ciphertext, which implies that the encapsulated key cannot be reconstructed if some tag of the ciphertext is punctured. In other words, it is desired that the share of the encryption key with respect to t cannot be recovered if $t \in T$. We observe that it resembles revocation system [30] and can be achieved by leveraging this well-studied cryptographic tool, in which the ciphertext under revocation list T cannot be decrypted whenever user t is revoked (i.e., $t \in T$). Following this way, we realize the puncture procedure by invoking the key generation algorithm of the revocation system. In particular, each time a new tag t is to be punctured, a random value msk_t is chosen for generating a corresponding secret key sk_t and subtracted from the master secret key msk (of the revocation system). Finally, the remaining part "$msk - \sum_t sk_t$" is (implicitly) used to produce a secret key sk_0 for a distinguished tag t_0, which is excluded in all punctures and the tag list of each ciphertext. To that end, we further refine the revocation system and introduce the concept of key-homomorphic revocation system with extended correctness that is crucial for our construction (including computing sk_0) and the security proof. For more details, please refer to Section 2.3 and Section 3.

1.2 Our Contributions

In this work, we present a modular way of constructing puncturable encryption inspired by the idea of distributed key-distributions. In particular, we first introduce a variant of identity-based revocation system, named *key-homomorphic*

Table 1. Comparison of Public-Key Puncturable Encryption Schemes

Schemes	Public key size	Secret key size	Ciphertext overhead	Punctured key size	Unbounded	
					punctures	ciphertext tags
	$(\lvert\mathbb{G}\rvert, \lvert\mathbb{G}_T\rvert, H)$	$\lvert\mathbb{G}\rvert$	$(\lvert\mathbb{G}\rvert, \lambda\text{-bit})$	$\lvert\mathbb{G}\rvert$	(Y/N, #)	(Y/N, #)
[26]	$(O(n), 1, 1)$	3	$(O(\hat{n}), 0)$	$3 \cdot (i+1)$	(Y, –)	(N, n)
$[19]_{\mathrm{IBE}}$	$(1, 0, k)$	m	$(1, k)$	$O(m)$	(N, d)	(N, 1)
$[18]_{\mathrm{IBBE}}$	$(O(k), 1, O(k))$	m	$(2, 1)$	$O(m)$	(N, d)	(N, 1)
Sec. 4.1	$(O(n), 1, 0)$	$O(n)$	$(2, 0)$	$O(n) \cdot (i+1)$	(Y, –)	(N, n)
Sec. 4.2	$(O(n), 1, 0)$	3	$(O(n), 0)$	$3 \cdot (i+1)$	(Y, –)	(N, n)
Sec. 4.3	$(5, 1, 0)$	3	$(O(\hat{n}), 0)$	$3 \cdot (i+1)$	(Y, –)	(Y, –)
Sec. 4.4	$(O(n), 2, 0)$	$O(n)$	$(6, 0)$	$O(n) \cdot (i+1)$	(Y, –)	(N, n)

$\lvert \cdot \rvert$: the bit-length of a group element, e.g., $\lvert\mathbb{G}\rvert$; H: a hash function; λ: a security parameter; $\hat{n}$: the number of tags attached to ciphertext; n: the upper-bound of $\hat{n}$ (i.e., $\lvert T\rvert \leq n$); d: the upper-bound on # of allowed punctures; i = # of tags associated with the current punctured key; $m = -d \cdot \ln p/(\ln 2)^2$ is the length of Bloom filter with false positive probability p [19].

Table 2. Comparison of Public-Key Puncturable Encryption Schemes

Schemes	Puncture	Encryption	Decryption*	Standard model	Negligible corr. error	Assumption
	(H, exp)	(pair, $\exp_T$, exp)	(pair, exp)			
[26]	$(1, O(n))$	$(0, 1, O(\hat{n}n))$	$(3, O(\hat{n})) \cdot (i+1)$	×	√	DBDH
$[19]_{\mathrm{IBE}}$	$(k, 0)$	$(k, k, 1)$	$(1, 0)$	×	×	BCDH
$[18]_{\mathrm{IBBE}}$	$(k, 0)$	$(0, 1, O(k))$	$(2, 2)$	×	×	GDDHE
Sec. 4.1	$(0, O(n))$	$(0, 1, O(\hat{n}))$	$(2, O(\hat{n})) \cdot (i+1)$	√	√	q-DBDHE
Sec. 4.2	$(0, O(n))$	$(0, 1, O(n))$	$(2, O(n)) \cdot (i+1)$	√	√	DBDH
Sec. 4.3	$(0, O(1))$	$(0, 1, O(\hat{n}))$	$(3, O(\hat{n})) \cdot (i+1)$	√	√	q-MEBDH
Sec. 4.4	$(0, O(n))$	$(0, 2, O(\hat{n}))$	$(6, O(\hat{n})) \cdot (i+1)$	√	√	DLIN

$\star$: decryption is done by a secret key punctured on i tags; $\hat{n}$: the number # of tags attached to ciphertext; n: the upper-bound of $\hat{n}$ (i.e., $\lvert T\rvert \leq n$); $k = -\log_2 p$ is the # of H's for a Bloom filter with false positive probability p [19].

identity-based revocable key encapsulation mechanism (KH-IRKEM) with extended correctness, and then propose a generic construction of *puncturable key encapsulation mechanism* (PKEM) from any such kind of KH-IRKEM scheme. Compared to the generic conversion of [19], our construction satisfies the standard correctness definition (i.e., negligible correctness error), enjoys more fine-grained revocation of decryption capability, and supports an unbounded number of punctures. Since the security and performance of our modular construction depends only on the underlying IRKEM scheme, our PKEM scheme can achieve the same level security as IRKEM without inducing additional security assumptions or computation redundancy. Based on the extensively-studied identity-based revocation systems, we also give four PKEM instantiations with distinct advantages, which are summarized as follows:

– Our first construction is the *first* PPE scheme that enjoys compact ciphertexts. Precisely, the ciphertext overhead consists of only two group elements.

Moreover, it has a faster encryption and decryption procedure (exactly decryption requires 33% less pairing computation) than the scheme by Green and Miers [26], and can be proven selectively secure in the standard model.
- Our second scheme has a comparable storage cost with [26]. Both schemes can be proven secure under the standard assumption —DBDH assumption, but our scheme enjoys more efficient encryption and decryption, especially when the number of tags encrypted is large. In more details, our encryption algorithm is independent of the number $\hat{n}$ of tags encrypted and the decryption requires 33% less pairing computation.
- Our third construction is proven secure under a stronger assumption, but features *compact* public key and *fast* puncture procedure, both of which depend not on the maximum number n of tags allowed per ciphertext. Moreover, it is the *first* scheme that has no constraint on the number of tags attached to ciphertext. It also enjoys a faster encryption algorithm compared to [26].
- As the first construction, our last scheme also features short ciphertexts, exactly consisting of six group elements. In contrast, it has a slightly slower encryption and decryption procedure, but can be proven adaptively secure based on the standard DLIN assumption, rather than a "q-type" one.

For more details on the comparison with previous works, please refer to Table 1 and Table 2 as well as the analysis given in Section 5.

2 Background

In this section, we give the notations used in this work and recollect the syntax and security of the relevant cryptographic primitives, such as public-key puncturable encryption and identity-based revocation system.

Notations. Security parameter is denoted by λ. For a finite set S, we let $s \xleftarrow{\$} S$ be the operation of sampling s uniformly at random from S. If S is a distribution, it denotes the operation of sampling s according to S. We write $a \leftarrow \mathrm{A}(\cdot)$ to denote the process of running algorithm $\mathrm{A}(\cdot)$ and assigning the result to a. If $\mathrm{A}(\cdot)$ is randomized, we use $\mathrm{A}(x; r)$ to denote the unique output of $\mathrm{A}(\cdot)$ taking as input x and randomness r. In addition, we denote by bold uppercase $\mathbf{A}$ (resp. lowercase $\boldsymbol{x}$) a matrix (resp. vector). Unless stated otherwise, all vectors are column vectors and row vectors are written as $\boldsymbol{x}^{\mathsf{T}}$. For two vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}_p^n$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$, we denote their inner product as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i$. For a matrix $\mathbf{A} = [a_{i,j}] \in \mathbb{Z}_p^{m \times n}$ and a group element $g \in \mathbb{G}$, we write $g^{\mathbf{A}}$ to denote the matrix $[g^{a_{i,j}}] \in \mathbb{G}^{m \times n}$. Also, we use $[a, b]$ to denote the set $\{a, a+1, \ldots, b-1, b\}$ for integers $b > a \geq 0$.

2.1 Bilinear Maps

We briefly review the relevant facts about bilinear maps. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be multiplicative cyclic groups of prime order p, and g, h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. An efficiently computable mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if it satisfies the following properties:

1. Bilinearity: for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: for $g \in \mathbb{G}_1, h \in \mathbb{G}_2$, $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

For matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_p^{m \times n}$, we let $e(g^{\mathbf{A}}, h^{\mathbf{B}}) = e(g, h)^{\mathbf{A}^\intercal \mathbf{B}}$ hereafter.

2.2 Puncturable Key-Encapsulation Mechanism

As mentioned in [19], a full-blown public-key puncturable encryption scheme can be generically converted from any puncturable key-encapsulation mechanism (PKEM), so next we only present the syntax and security of PKEM. Following the definition given in [19], a PKEM scheme with key space $\mathcal{K}$ and tag space $\mathcal{T}$ generally consists of four polynomial time algorithms (KeyGen, Enc, Punc, Dec) with the specifications below:

- $\mathsf{KeyGen}(1^\lambda, n)$ takes as input a security parameter λ and a maximum number n of tags allowed for each ciphertext, and outputs a public and secret key pair (PK, SK). Note that $n \in \mathbb{N} \cup \{\infty\}$ and "∞" means the number of tags per ciphertext is unbounded.
- $\mathsf{Enc}(PK, T)$ takes as input a public key PK and a set of tags T such that $|T| \leq n$, and outputs an encapsulated key K and a ciphertext CT.
- $\mathsf{Punc}(SK_{i-1}, t)$ takes as input a secret key SK_{i-1} and a tag t, where $SK_0 = SK$, and outputs a new secret key SK_i that can decrypt what SK_{i-1} can except for those encrypted under tag t.
- $\mathsf{Dec}(SK_i, CT, T)$ takes as input a secret key SK_i and a ciphertext CT generated under a list of tags T, and outputs the encapsulated key K or $\perp$ (the latter indicates the decapsulation fails).

Definition 1 (Correctness). *For all $\lambda \in \mathbb{N}$, $n \in \mathbb{N} \cup \{\infty\}$, and $T \subseteq \mathcal{T}$ such that $|T| \leq n$, let $(PK, SK) \leftarrow \mathsf{KeyGen}(1^\lambda, n)$ and $(K, CT) \leftarrow \mathsf{Enc}(PK, T)$, then we have that $\mathsf{Dec}(SK, CT, T) = K$. Moreover, for any ℓ times of invoking $SK_i \leftarrow \mathsf{Punc}(SK_{i-1}, t')$ such that $t' \notin T$, it holds that*

$$\Pr[\mathsf{Dec}(SK_\ell, CT, T) = \perp] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of all algorithms.

Remark 1. Our syntax is slightly different from [19]. In particular, our encryption algorithm also takes as input a list of tags T instead of only PK. Thus, our puncture algorithm is operated on tag t rather than ciphertext CT. In this way, many ciphertexts under the same tag t (e.g., all ciphertexts from the same sender) can be revoked by executing the puncture algorithm once. In fact, our PKEM is more similar to the key encapsulation version of public-key puncturable encryption initialized by Green and Miers [26], which enjoys fine-grained revocation of decryption capability.

The security of PKEM is adapted from that of PPE in [26]. It is defined via an IND-PUN-ATK game, which incorporates both CPA and CCA variants. The game is played between a challenger and an adversary as follows.

Setup: On input a security parameter λ and a maximum number n of tags allowed per ciphertext, the challenger runs $(PK, SK) \leftarrow \mathsf{KeyGen}(1^\lambda, n)$. Then it returns PK and initializes two empty sets $\mathtt{P}$, $\mathtt{C}$ and a counter $i = 0$.

Phase 1: The adversary adaptively issues the following queries

- **Puncture**(t'): On input a tag t', the challenger increments counter i, computes $SK_i \leftarrow \mathsf{Punc}(SK_{i-1}, t')$ and adds t' to $\mathtt{P}$.
- **Corrupt**(): The first time the adversary issues this query, the challenger returns the most recent secret key SK_i and sets $\mathtt{C} \leftarrow \mathtt{P}$. For subsequent queries, directly returns $\perp$.
- **Decrypt**(CT, T): On input a ciphertext CT and the associated tags T, the challenger returns $K \leftarrow \mathsf{Dec}(SK_i, CT, T)$ if ATK = CCA, otherwise returns $\perp$.

Challenge: On input challenge tags $T^* \subseteq \mathcal{T}$, the challenger directly rejects if the adversary has previously issued a **Corrupt** query and $T^* \cap \mathtt{C} = \emptyset$. Otherwise, it picks $b \xleftarrow{\$} \{0,1\}, K_1 \xleftarrow{\$} \mathcal{K}$ and computes $(K_0, CT^*) \leftarrow \mathsf{Enc}(PK, T^*)$. At last, it returns (K_b, CT^*) to the adversary.

Phase 2: This phase is the same as Phase 1 except for the following restrictions

- **Corrupt**(): Returns $\perp$ if $T^* \cap \mathtt{P} = \emptyset$.
- **Decrypt**(CT, T): Returns $\perp$ if $(CT, T) = (CT^*, T^*)$.

Guess: The adversary outputs a guess b' and wins the game if $b' = b$.

Definition 2 (Adaptive Security). *A PKEM scheme* $\mathsf{PKEM} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Punc}, \mathsf{Dec})$ *is IND-PUN-ATK secure for ATK* $\in \{CPA, CCA\}$ *if for all probabilistic polynomial time (PPT) adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *winning in the IND-PUN-ATK game is*

$$\mathsf{Adv}^{\text{IND-PUN-ATK}}_{\mathcal{A},\mathsf{PKEM}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{negl}(\lambda)$ *is a negligible function of* λ*.*

We also define a weak security named selective security by an IND-sPUN-ATK game. It is similar to the above game except that the adversary is required to submit the challenge tag list $T^* \subseteq \mathcal{T}$ before the setup phase.

Definition 3 (Selective Security). *A PKEM scheme* $\mathsf{PKEM} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Punc}, \mathsf{Dec})$ *is IND-sPUN-ATK secure for ATK* $\in \{CPA, CCA\}$ *if for all PPT adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *winning in the IND-sPUN-ATK game is*

$$\mathsf{Adv}^{\text{IND-sPUN-ATK}}_{\mathcal{A},\mathsf{PKEM}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

2.3 Key-Homomorphic Identity-Based Revocation Mechanism

In this part, we first recall the syntax and security of identity-based revocation scheme, and then introduce a new concept — *key-homomorphic identity-based revocable key encapsulation mechanism* (KH-IRKEM) — for our application.

In fact, we do not need a full-blown revocation encryption scheme. Instead, an identity-based revocable key encapsulation mechanism (IRKEM) is sufficient for our application, where an encapsulated key can be recovered by a receiver if and only if s/he is not revoked during the encapsulation phase.

More formally, an IRKEM scheme with master secret key space $\mathcal{MSK}$, private key space $\mathcal{SK}$, encapsulated key space $\mathcal{K}$ and identity space $\mathcal{ID}$ is comprised of a tuple of polynomial time algorithms (Params, MKGen, KeyExt, Enc, Dec):

- $\mathsf{Params}(1^\lambda, n)$ takes as input a security parameter λ and a maximum number n of revoked users, and outputs system parameters pp that is (implicitly) taken as an additional input of the rest algorithms. Note that $n \in \mathbb{N} \cup \{\infty\}$, and "$\infty$" indicates the number of revoked users is unbounded.
- $\mathsf{MKGen}(pp)$ takes as input public parameters pp and outputs a master public key mpk and a master secret key msk.
- $\mathsf{KeyExt}(msk, id)$ takes as input a master secret key msk and an identity id, outputs a private key sk_{id} for the identity id. When this algorithm is randomized, the associated random coin space is assumed to be $\mathcal{R}$.
- $\mathsf{Enc}(mpk, R)$ takes as input a master public key mpk and a list R of revoked users, where $|R| \leq n$, and outputs a symmetric key k and a ciphertext ct, such that any user with private key sk_{id} for $id \notin R$ can recover the encapsulated key k.
- $\mathsf{Dec}(sk_{id}, id, ct, R)$ takes as input a private key sk_{id} for an identity id and a ciphertext ct associated with the revocation list R, and outputs an encapsulated key k if $id \notin R$ and $\perp$ otherwise.

Definition 4 (Correctness). *For all $\lambda \in \mathbb{N}, n \in \mathbb{N} \cup \{\infty\}$, $R \subseteq \mathcal{ID}$ such that $|R| \leq n$, let $pp \leftarrow \mathsf{Params}(1^\lambda, n)$, $(mpk, msk) \leftarrow \mathsf{MKGen}(pp)$, $(k, ct) \leftarrow \mathsf{Enc}(mpk, R)$, and $sk_{id} \leftarrow \mathsf{KeyExt}(msk, id)$ for $id \notin R$, it holds that*

$$\Pr[\mathsf{Dec}(sk_{id}, id, ct, R) = k] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness of the associated algorithms.

Remark 2. Similar to the definition of IBE in [5,6], we add a parameter generation algorithm to the specification of IRKEM, in order to make the public parameters explicitly distinct from the master public key. This implies that the parameters may not depend on the master secret key, although the master public key might. In this work, the parameters may include the description of groups, group generators and the like. For simplicity, we additionally assume that, as in [6], the master secret key is randomly drawn from $\mathcal{MSK}$ and the master public key is derived deterministically/probabilistically from it.

The security of IRKEM is defined by an IND-RL-ATK game, which incorporates both CPA and CCA variants. The game is played between a challenger and an adversary, which is described as follows.

Setup: On input a security parameter λ and a maximum number n of revoked users, the challenger generates $pp \leftarrow \mathsf{Params}(1^\lambda, d)$ and $(mpk, msk) \leftarrow \mathsf{MKGen}(pp)$, then returns pp, mpk and initializes an empty set $\mathcal{Q}$.

Phase 1: The adversary can adaptively issue the following queries

- **Key Extract**(id): On input an identity id, the challenger returns a corresponding private key $sk_{id} \leftarrow \mathsf{KeyExt}(msk, id)$ and adds id to $\mathcal{Q}$.
- **Decrypt**(id, ct, R): On input an identity id, a ciphertext ct and the associated revocation list R, the challenger computes sk_{id} and returns $k \leftarrow \mathsf{Dec}(sk_{id}, id, ct, R)$ if ATK = CCA, otherwise returns $\perp$.

Challenge: On input a list of revoked identities $R^* \subseteq \mathcal{ID}$, the challenger directly rejects if $\mathcal{Q} \setminus R^* \neq \emptyset$. Otherwise, it picks $b \xleftarrow{\$} \{0,1\}, k_1 \xleftarrow{\$} \mathcal{K}$ and computes $(k_0, ct^*) \leftarrow \mathsf{Enc}(mpk, R^*)$. Finally, it sends (k_b, ct^*) back to the adversary.

Phase 2: This is the same as Phase 1 except with below restrictions

- **Key Extract**(id): Returns $\perp$ if $id \notin R^*$.
- **Decrypt**(id, ct, R): Returns $\perp$ if $(ct, R) = (ct^*, R^*)$.

Guess: The adversary outputs a guess b' and wins the game if $b' = b$.

Definition 5 (Adaptive Security). *An IRKEM scheme $\Sigma = (\mathsf{Params}, \mathsf{MKGen}, \mathsf{KeyExt}, \mathsf{Enc}, \mathsf{Dec})$ is IND-RL-ATK secure for ATK $\in \{CPA, CCA\}$ if for all $\lambda \in \mathbb{N}$ and PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ winning in the IND-RL-ATK game is*

$$\mathsf{Adv}^{\text{IND-RL-ATK}}_{\mathit{IRKEM},\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

Similar to the selective security of PKEM, we also define a selective security for IRKEM by an IND-sRL-ATK game.

Definition 6 (Selective Security). *An IRKEM scheme $\Sigma = (\mathsf{Params}, \mathsf{MKGen}, \mathsf{KeyExt}, \mathsf{Enc}, \mathsf{Dec})$ is IND-sRL-ATK secure for ATK $\in \{CPA, CCA\}$ if for all $\lambda \in \mathbb{N}$ and PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ winning in the IND-sRL-ATK game is*

$$\mathsf{Adv}^{\text{IND-sRL-ATK}}_{\mathit{IRKEM},\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

Next we introduce the additional properties of IRKEM, desired for our applications. The first is called Extended Correctness. Informally, this property ensures that a legally encapsulated key can be computed correctly in an alternative way. It is formalized in Definition 7. To formally define this property, we will write the random coin explicitly in the encapsulation algorithm.

Definition 7 (Extended Correctness). *For all $\lambda \in \mathbb{N}, n \in \mathbb{N} \cup \{\infty\}$, $R_i \subseteq \mathcal{ID}$ such that $|R_i| \leq n$, any $pp \leftarrow \mathsf{Params}(1^\lambda, n)$, $(mpk_i, msk_i) \leftarrow \mathsf{MKGen}(pp)$, $(k_i, ct_i) = \mathsf{Enc}(mpk_i, R_i; s_i)$, and $sk_i \leftarrow \mathsf{KeyExt}(msk_i, id_i)$ for $i \in \{1, 2\}$, we let $(\widehat{k}, \widehat{ct}) = \mathsf{Enc}(mpk_1, R_2; s_2)$. Then an IRKEM scheme Σ is called extended correct if for $id_1 \notin R_2$ it satisfies that*

$$\Pr[\mathsf{Dec}(sk_1, id_1, ct_2, R_2) = \widehat{k}] \geq 1 - \mathsf{negl}(\lambda).$$

We note that the encapsulated key $\hat{k}$ can be correctly recovered by $\hat{k} \leftarrow \mathsf{Dec}(sk_1, id_1, \widehat{ct}, R_2)$ in terms of the standard correctness (cf. Definition 4). Here, it is further required that $\hat{k}$ can be obtained by "decapsulating" other ciphertexts generated under the same revocation list and random coins. Alternatively, this property means decapsulating a ciphertext with a mismatched private key can produce a legitimate encapsulated key. Thus we are able to compute a legally encapsulated key in a different way than by running the encapsulation algorithm.

Hereafter, when the correctness is mentioned, it refers to both the regular and the extended correctness, unless stated otherwise. Now we continue to define the second property — key-homomorphism, which is critical for distributing the encapsulated key. More specifically, this property should hold with respect to both the encapsulated key and the private key, as formalized below.

Definition 8 (Key-Homomorphism). *Let $\Sigma = (\mathsf{Params}, \mathsf{MKGen}, \mathsf{KeyExt}, \mathsf{Enc}, \mathsf{Dec})$ be an IRKEM scheme. We assume that the randomness space $\mathcal{R}$ (associated with $\mathsf{KeyExt}(\cdot)$ if it is randomized) and key spaces $\mathcal{MSK}, \mathcal{SK}$ and $\mathcal{K}$ form four groups $(\mathcal{R}, *), (\mathcal{MSK}, +), (\mathcal{SK}, \otimes)$ and $(\mathcal{K}, \odot)$. Moreover, we assume that the encapsulated key is in the form of $f(msk, s)$, where $s \in \mathcal{S}$ is the random coin consumed in the encapsulation algorithm, i.e., $\big(k = f(msk, s), ct\big) = \mathsf{Enc}(mpk, R; s)$. Then the IRKEM scheme Σ is called key-homomorphic if it satisfies the above correctness, and fulfills the following conditions for all $id \in \mathcal{ID}$, $msk, msk' \in \mathcal{MSK}$, $r, r' \in \mathcal{R}$ and $s \in \mathcal{S}$:*

1. $\mathsf{KeyExt}(msk, id; r) \otimes \mathsf{KeyExt}(msk', id; r') = \mathsf{KeyExt}(msk + msk', id;\ r * r')$,
2. $f(msk, s) \odot f(msk', s) = f(msk + msk', s)$.

This property plays an important role in our work, which reflects in both the construction and the security analysis. We remark that if an IRKEM Σ is correct in the sense of Definition 7 and secure (either selectively or adaptively), then $f(msk, s)$ associated with the second property of Definition 8 should depend on msk non-trivially. Otherwise, if $f(msk, s) = f'(s)$ is independent of msk and Σ is extended correct, then an adversary can break the security of Σ as follows. After receiving the master public key mpk_2 and the challenge ciphertext ct_2 and encapsulated key k_2, she generates a new (msk_1, mpk_1), chooses an $id_1 \notin R_2$ and computes $sk_1 \leftarrow \mathsf{KeyExt}(msk_1, id_1)$. Then she can recover $k_2 = \hat{k}$ by using the extended correctness property and break the security easily. Examples of such schemes can be derived from transferring the lattice-based NIPEs (the first two constructions) of [29] to IRKEMs by changing the encryption/decryption functions to encapsulation/decapsulation functions similar to Subsections 4.1-4.3. It is then easy to see that $f(msk, s)$ is independent of msk. This implies that the derived IRKEMs are not extended correct as they are proven to be secure in [29].

In the following, we first propose a generic construction of PKEM from any KH-IRKEM scheme with extended correctness, and then present several instantiations with distinct features.

3 Construction of PKEM from KH-IRKEM

In this section, we present a generic construction of PKEM from any KH-IRKEM scheme with extended correctness, as defined before. Let $\Sigma = (\mathsf{IR.Params}, \mathsf{IR.MKGen}, \mathsf{IR.KeyExt}, \mathsf{IR.Enc}, \mathsf{IR.Dec})$ be a KH-IRKEM scheme with identity space $\mathcal{ID}$, then a PKEM scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Punc}, \mathsf{Enc}, \mathsf{Dec})$ with tag space $\mathcal{T} = \mathcal{ID}$ is constructed from Σ as follows.

- $\mathsf{KeyGen}(1^\lambda, n)$: On input a parameter λ and an index $n \in \mathbb{N} \cup \{\infty\}$, it first generates $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$. Then it selects a distinguished tag $t_0 \in \mathcal{T}$, which will never be punctured and encrypted later, and produces $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk, t_0)$. Finally, it outputs the public and secret key pair
$$(PK, SK) = \big((pp, mpk), (sk_0, t_0)\big).$$
- $\mathsf{Punc}(SK_{i-1}, t_i)$: On input a punctured secret key $SK_{i-1} = \big((sk_0, t_0), \ldots, (sk_{i-1}, t_{i-1})\big)$ for tags $\{t_\ell\}_{\ell=1}^{i-1}$ and a tag $t_i \in \mathcal{T} \setminus \{t_0\}$, where $SK_0 = SK$, it randomly chooses $msk_i \in \mathcal{MSK}$ [2] and produces a new puncture secret key SK_i for $\{t_\ell\}_{\ell=1}^{i}$ as below:
  1. Computes $sk_0' = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$ and $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$, where sk_0 in SK_{i-1} is updated to sk_0'.
  2. Sets $SK_i = \big((sk_0', t_0), (sk_1, t_1), \ldots, (sk_{i-1}, t_{i-1}), (sk_i, t_i)\big)$, where sk_j for all $j \in [1, i-1]$ remains identical to SK_{i-1}.
- $\mathsf{Enc}(PK, T)$: On input public key $PK = (pp, mpk)$ and a list of tags T such that $|T| \le n$ and $T \subseteq \mathcal{T} \setminus \{t_0\}$, it computes
$$\big(f(msk, s), ct\big) = \mathsf{IR.Enc}(mpk, T; s)$$
and outputs $(K, CT) = (f(msk, s), ct)$ along with T.
- $\mathsf{Dec}(SK_i, CT, T)$: On input a punctured secret key $SK_i = \big((sk_0, t_0), \ldots, (sk_i, t_i)\big)$ and a ciphertext CT along with tags T, it returns $\perp$ if there exists $j \in [1, i]$ such that $t_j \in T$. Otherwise, it recovers the encapsulated key as:
  1. Computes $k_j = \mathsf{IR.Dec}(sk_j, t_j, CT, T)$ for all $j \in [0, i]$.
  2. Calculates $K' = \bigodot_{j=0}^{i} k_j$ and outputs K'.

The correctness follows from the (extended) correctness of the underlying KH-IRKEM scheme and its key-homomorphic properties. To be more precise, we assume that $(K, CT) = \mathsf{IR.Enc}(mpk, T; s)$ and $sk_j \leftarrow \mathsf{IR.KeyExt}(msk_j, t_j)$ such that $t_j \notin T$ for all $j \in [1, i]$. The key-homomorphic property of $\mathsf{IR.KeyExt}(\cdot, \cdot)$ indicates the current key component sk_0 (of SK_i) is in the form of $\mathsf{IR.KeyExt}(msk - \sum_{j=1}^{i} msk_j, t_0)$. Then we have that

[2] Recall that the master secret key is assumed to be randomly drawn from $\mathcal{MSK}$, as remarked in Section 2.3.

– $k_0 = \mathsf{IR.Dec}(sk_0, t_0, CT, T) = f(msk - \sum_{j=1}^{i} msk_j, s)$, and
– $k_j = \mathsf{IR.Dec}(sk_j, t_j, CT, T) = f(msk_j, s)$ for all $j \in [1, i]$,

where the second equalities derive from the extended correctness of KH-IRKEM scheme Σ. Finally, the key-homomorphism of $f(\cdot, \cdot)$ yields that

$$K' = f\big(msk - \sum_{j=1}^{i} msk_j, s\big) \odot \bigodot_{j=1}^{i} f(msk_j, s) = f(msk, s).$$

We remark that the ciphertext CT taken in decapsulation process is generated under mpk, while the private keys sk_j are computed from new master secret keys msk_j rather than msk. In this case, the standard correctness is insufficient, and hence the extended correctness is crucial for the correctness of our PKEM.

3.1 Security Analysis

We first show that the proposed PKEM scheme is IND-sPUN-CPA secure if the underlying IRKEM scheme is IND-sRL-CPA secure. Then we further discuss its adaptive security based on the adaptive security of the KH-IRKEM scheme.

Theorem 1. *The proposed generic construction PKEM is IND-sPUN-CPA secure, if the underlying IRKEM scheme is key-homomorphic and IND-sRL-CPA secure. More precisely, for any PPT adversary $\mathcal{A}$ against the security of our PKEM scheme, it holds that*

$$\mathrm{Adv}^{\text{IND-sPUN-CPA}}_{\mathit{PKEM},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\text{IND-sRL-CPA}}_{\mathit{IRKEM},\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ is some PPT algorithm against the security of the IRKEM scheme.

Proof. The proof is conducted through a sequence of games that starts with the real IND-sPUN-CPA game and ends with a game in which the adversary has a negligible advantage. Moreover, each two successive games are shown to be (computationally) indistinguishable. Hereafter, we let Win_i denote the event that the adversary $\mathcal{A}$ wins in game G_i. For sake of clarity, we assume that $\mathcal{A}$ makes at most q puncture queries, say $\{t_1, t_2, \ldots, t_q\}$, and at least one of them, say t_i for some $i \in [1, q]$, belongs to the set of challenge tags T^*. It is also assumed that, without loss of generality, the corrupt query is made after all q punctures. Then the current punctured secret key is sent back to $\mathcal{A}$ directly.

Game G_0: It is the real game played between a challenger and an adversary $\mathcal{A}$, as described in Section 2.2. In more details, $\mathcal{A}$ first submits a set of challenge tags T^* such that $|T^*| \leq n$. After that, the challenger chooses a distinguished tag $t_0 \in \mathcal{T}$ and runs $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$, $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$ and $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk, t_0)$ to produce the public and secret key pair $(PK, SK) = \big((pp, mpk), (sk_0, t_0)\big)$. In addition, it initializes an empty set $\mathtt{P}$ for keeping track of puncture queries. Then it returns PK to the adversary $\mathcal{A}$, and answers the puncture queries and the challenge query as follows:

– **Puncture**(t_i): The challenger chooses $msk_i \xleftarrow{\$} \mathcal{MSK}$, computes $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ and updates the first component sk_0 of SK_{i-1} as $sk_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$. Then it sets $SK_i = \big((sk_0, t_0), \ldots, (sk_{i-1}, t_{i-1}), (sk_i, t_i)\big)$, where $SK_0 = SK$, and records t_i to $\mathtt{P}$. Finally, it returns $SK_q = \big((sk_0, t_0), (sk_1, t_1), \ldots, (sk_q, t_q)\big)$ to $\mathcal{A}$ after all q puncture queries.
– **Challenge**: On input the challenge $T^* \subseteq \mathcal{T} \setminus \{t_0\}$ [3], the challenger computes $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$ and randomly chooses $K_1^* \xleftarrow{\$} \mathcal{K}$. Then it selects $b \xleftarrow{\$} \{0,1\}$ and outputs (K_b^*, CT^*).

Eventually, the adversary $\mathcal{A}$ outputs a guess b', and wins the game if $b' = b$. We get from the security definition of PKEM (cf. Definition 2) that

$$\mathsf{Adv}^{\text{IND-sPUN-CPA}}_{\mathsf{PKEM},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Win}_0] - \frac{1}{2}\right|.$$

Game G_1: This game is identical to G_0, except that the master secret keys $msk_1, msk_2, \ldots, msk_q \in \mathcal{MSK}$ are sampled beforehand instead of on-the-fly and used straightforwardly to simulate the puncture queries. In particular, all queries are answered as follows:

– **Puncture**(t_i): The challenger computes $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ and updates the first component of SK_{i-1} as $sk_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$, by directly using msk_i chosen before. After that, it sets $SK_i = \big((sk_0, t_0), \ldots, (sk_{i-1}, t_{i-1}), (sk_i, t_i)\big)$ and adds t_i to $\mathtt{P}$. Finally, it returns $SK_q = \big((sk_0, t_0), (sk_1, t_1), \ldots, (sk_q, t_q)\big)$ to $\mathcal{A}$ after all q puncture queries.
– **Challenge**: On input the challenge $T^* \subseteq \mathcal{T} \setminus \{t_0\}$, the challenger computes $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$ and randomly picks $K_1^* \xleftarrow{\$} \mathcal{K}$. Then it outputs (K_b^*, CT^*) where b is chosen uniform randomly from $\{0,1\}$.

At last, the adversary $\mathcal{A}$ outputs her guess b'. It can be seen from the above that the way of sampling the master secret keys does not change the view of the adversary. Therefore, it holds that

$$\Pr[\mathsf{Win}_1] = \Pr[\mathsf{Win}_0].$$

Game G_2: It is the same as above game except that the component sk_0 of the finally corrupted secret key SK_q is generated in a different way. Briefly, sk_0 here is generated in a direct manner rather than by sequential updates (i.e., $sk_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$). More specifically, after receiving challenge tags T^* the challenger runs $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$, and picks in advance $msk_1, msk_2, \ldots, msk_q \in \mathcal{MSK}$ that will be used to answer the puncture queries issued by $\mathcal{A}$ later. Then it sets $msk_0 = msk - \sum_{i=1}^{q} msk_i$ and computes $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk_0, t_0)$ for the distinguished tag $t_0 \in \mathcal{T}$. After that, it returns $PK = (pp, mpk)$ and simulates the puncture queries and challenge query as below:

[3] We always assume that $T^* \cap \mathtt{P} \neq \emptyset$, otherwise it will be rejected according to the security definition.

– **Puncture**(t_i): The challenger uses msk_i (chosen above) to compute $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ for the i-th puncture query t_i, and records t_i to $\mathtt{P}$. After receiving all q puncture queries from $\mathcal{A}$, it returns $SK_q = \big((sk_0, t_0), \ldots, (sk_q, t_q)\big)$. Recall that sk_0 is generated at the beginning.
– **Challenge**: On input the challenge T^* issued by $\mathcal{A}$, the challenger computes $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$ and chooses $K_1^* \xleftarrow{\$} \mathcal{K}$. Then it chooses a random bit $b \in \{0,1\}$ and returns (K_b^*, CT^*).

Finally, $\mathcal{A}$ outputs her guess b'. Clearly, the distribution of this game is identical to G_1, so we have

$$\Pr[\mathsf{Win}_2] = \Pr[\mathsf{Win}_1].$$

Game G_3: In this game, we assume that, without loss of generality, the j-th puncture query t_j is the first tag belonging to the set T^* of challenge tags. Notice that, there exists at least one puncture query contained in T^* in terms of the security definition of PKEM (cf. Definition 2), and it is easy to find the index j given T^*. Then the difference of this game from G_2 is the way of generating sk_0 and sk_j (associated with t_j).

In particular, the challenger in this game runs $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$, chooses $msk_0, \ldots, msk_{j-1}, msk_{j+1}, \ldots, msk_q \in \mathcal{MSK}$ uniformly at random, and sets $msk_j = msk - \sum_{i=0,\neq j}^{q} msk_i$. Then it uses msk_0 to compute $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk_0, t_0)$ for the distinguished tag $t_0 \in \mathcal{T}$ and uses msk_i to compute $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ for the i-th puncture query t_i, where $i \in [1, q]$. As for the challenge query, it is simulated in the same way as before.

It is not difficult to see $\mathcal{A}$'s views in G_2 and G_3 are identical, as they rely essentially on the identical distributions $(msk, msk_0 = msk - \sum_{i=1}^{q} msk_i, msk_1, \ldots, msk_q)$ and $(msk, msk_0, \ldots, msk_{j-1}, msk_j = msk - \sum_{i=1,\neq j}^{q} msk_i, msk_{j+1}, \ldots, msk_q)$, respectively. Therefore, we get that

$$\Pr[\mathsf{Win}_3] = \Pr[\mathsf{Win}_2].$$

Now, what remains to do is to show the advantage of $\mathcal{A}$ winning in G_3 is negligible in λ. It is formally stated as the following lemma.

Lemma 1. *Provided that the underlying IRKEM scheme Σ is IND-sRL-CPA secure and key-homomorphic, then the advantage of $\mathcal{A}$ winning in G_3 is negligible in λ. That is,*

$$\left|\Pr[\mathsf{Win}_3] - \frac{1}{2}\right| = \mathsf{Adv}^{\text{IND-sRL-CPA}}_{\mathsf{IRKEM},\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ is some PPT algorithm against the security of the IRKEM scheme.

Proof (of Lemma 1). Suppose for sake of contradiction that there is an efficient adversary $\mathcal{A}$ winning in G_3 with non-negligible advantage, then we can find an efficient algorithm $\mathcal{B}$ that succeeds to break the IND-sRL-CPA security of the underlying IRKEM scheme Σ as follows.

After receiving the set T^* of challenge tags from $\mathcal{A}$, $\mathcal{B}(1^\lambda)$ sets it as his own challenge and submits T^* to the challenger of the IRKEM scheme. Then $\mathcal{B}$ returns to $\mathcal{A}$ the response pp and mpk, such that $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$. After that, $\mathcal{B}$ chooses uniformly at random $msk_0, \ldots, msk_{j-1}, msk_{j+1}, \ldots, msk_q$ from $\mathcal{MSK}$, and uses msk_0 to compute $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk_0, t_0)$ for the distinguished tag $t_0 \in \mathcal{T}$ chosen by himself. Then $\mathcal{B}$ proceeds to simulate the puncture queries and the challenge query as follows:

- **Puncture**(t_i): For the i-th puncture query t_i, $\mathcal{B}$ directly uses msk_i chosen above to generate $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ if $i \neq j$. Otherwise, $\mathcal{B}$ forwards $t_j \in T^*$ to the key extraction oracle of the IRKEM scheme and gets the corresponding private key sk'_j. Then $\mathcal{B}$ computes $sk_j = sk'_j \otimes \mathsf{IR.KeyExt}(-\sum_{i=0,\neq j}^{q} msk_i, t_j)$. In addition, $\mathcal{B}$ adds t_i to $\mathtt{P}$. Once finishing the simulation of all q puncture queries, $\mathcal{B}$ returns $SK_q = \big((sk_0, t_0), (sk_1, t_1), \ldots, (sk_q, t_q)\big)$ to $\mathcal{A}$.
- **Challenge**: $\mathcal{B}$ gets the response (to the challenge T^*) from the challenger of the IRKEM scheme. In particular, the response is (K_b^*, CT^*), such that $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$, $K_1^* \xleftarrow{\$} \mathcal{K}$, and $b \xleftarrow{\$} \{0,1\}$. Then $\mathcal{B}$ outputs (K_b^*, CT^*) to the adversary $\mathcal{A}$.

At last, $\mathcal{B}$ outputs what $\mathcal{A}$ outputs. From the above, we can see that $\mathcal{B}$ perfectly simulates G_3, hence we have

$$\mathsf{Adv}^{\text{IND-sRL-CPA}}_{\mathsf{IRKEM},\mathcal{B}}(\lambda) = \left|\Pr[\mathcal{B}(1^\lambda, \mathsf{View}) = b] - \frac{1}{2}\right| = \left|\Pr[\mathsf{Win}_3] - \frac{1}{2}\right|,$$

where View is the view of $\mathcal{B}$ in the IRKEM game that consists of pp, mpk, sk'_j and (K_b^*, CT^*).

Putting all above equations together, we get the advantage of any PPT adversary $\mathcal{A}$ against our PKEM scheme

$$\mathsf{Adv}^{\text{IND-sPUN-CPA}}_{\mathsf{PKEM},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Win}_3] - \frac{1}{2}\right| = \mathsf{Adv}^{\text{IND-sRL-CPA}}_{\mathsf{IRKEM},\mathcal{B}}(\lambda).$$

Theorem 2. *The proposed generic construction PKEM is IND-PUN-CPA secure, if the underlying IRKEM scheme is key-homomorphic and IND-RL-CPA secure. More precisely, for any PPT adversary $\mathcal{A}$ against the security of our PKEM scheme, it holds that*

$$\mathrm{Adv}^{\text{IND-PUN-CPA}}_{\mathit{PKEM},\mathcal{A}}(\lambda) \leq q \cdot \mathsf{Adv}^{\text{IND-RL-CPA}}_{\mathit{IRKEM},\mathcal{B}}(\lambda),$$

where q is the maximum number of puncture queries issued by $\mathcal{A}$ and $\mathcal{B}$ is some PPT algorithm against the security of the IRKEM scheme.

Proof (Sketch). To show the adaptive security of our PKEM scheme, we only need to guess which puncture query is belonging to the set of challenge tags T^* in the previous proof. The probability of guessing it correctly is at least $1/q$, assuming that the upper-bound on the number of puncture queries issued by $\mathcal{A}$ is q. For the detailed proof, please refer to Appendix A.

4 Instantiations of KH-IRKEM

In this section, we present several concrete IRKEM schemes derived from existing Identity-Based Revocation (IBR) schemes or Non-zero Inner Product Encryption (NIPE) schemes, and show that they satisfy the desired properties for our purpose. Particularly, the design of IRKEM schemes from the NIPE schemes follows the Embedding Lemma (see Proposition 1 in [3]), and thus the security of the IRKEM schemes can be reduced to the NIPE schemes. Then by applying our generic construction in Section 3, we obtain the first PKEM schemes that not only support unbounded number of punctures, but also features constant-size ciphertext, short public keys, or unbounded number of tags per ciphertext.

4.1 KH-IRKEM with Compact Ciphertexts

The first IRKEM scheme is derived from the NIPE scheme in [38], which is proven secure under the n-DBDHE assumption below.

q-DBDHE Assumption. Let $(\mathbb{G}, \mathbb{G}_T)$ be cyclic groups of prime order p with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The q-Decision Bilinear Diffie-Hellman Exponent (n-DBDHE) problem is, given

$$\big(g, g^a, g^{(a^2)}, \ldots, g^{(a^q)}, g^{(a^{q+2})}, \ldots, g^{(a^{2q})}, h, T\big)$$

where $a \xleftarrow{\$} \mathbb{Z}_p, g, h \xleftarrow{\$} \mathbb{G}$ and $T \in \mathbb{G}_T$, to decide if $T = e(g,h)^{a^{q+1}}$ or if T is randomly chosen from $\mathbb{G}_T$.

Description. This scheme consists of five polynomial-time algorithms (Params, MKGen, $\mathsf{KeyExt}, \mathsf{Enc}, \mathsf{Dec}$) with the following specifications:

- $\mathsf{Params}(1^\lambda, n)$: The algorithm takes a security parameter λ and an integer $n \in \mathbb{N}$, and generates a pair of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with bilinear map e. Then it randomly chooses $\beta, b_1, \ldots, b_n \in \mathbb{Z}_p$ and $g \in \mathbb{G}$, and computes $h = g^\beta$ and $h_i = g^{b_i}$ for all $i \in [1, n]$. Finally, it outputs the public parameters
$$pp = \big((\mathbb{G}, \mathbb{G}_T, e), g, h, \{h_i\}_{i\in[1,n]}\big).$$
- $\mathsf{MKGen}(pp)$: Given the public parameters pp, it chooses $\alpha \in \mathbb{Z}_p$ uniformly at random, and then computes and outputs the master secret key and master public key pair
$$(msk, mpk) = \big(\alpha, e(g,g)^\alpha\big).$$
- $\mathsf{KeyExt}(msk, id)$: Given a master secret key $msk = \alpha$ and an identity $id \in \mathbb{Z}_p$, this algorithm first defines a vector $\boldsymbol{x}_{id} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ such that $x_i = id^{i-1}$ for all $i \in [1, n]$, then it chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the private key $sk_{id} = (d_1, d_2, k_1, \ldots, k_n) \in \mathbb{G}^{n+2}$ as
$$d_1 = g^\alpha h_1^r,\ d_2 = g^r,\ k_1 = h^r,\ k_i = \big(h_1^{-x_i} h_i\big)^r \text{ for } \forall i \in [2, n].$$

– $\mathsf{Enc}(mpk, R)$: Given a master public key mpk and a revocation list $R = \{id_1, id_2, \ldots, id_m\}$ such that $m < n$, the algorithm generates the encapsulated key $k \in \mathbb{G}_T$ and ciphertext $ct = (c_1, c_2) \in \mathbb{G}^2$ as follows:
  1. Define a vector $\boldsymbol{y}_R = (y_1, \ldots, y_n)$, where $\{y_i\}_{i\in[1,m+1]}$ are the coefficients of the polynomial $f_R(z) = \prod_{id_j \in R}(z - id_j) = \sum_{i=1}^{m+1} y_i \cdot z^{i-1}$, and all other coordinates $\{y_i\}_{i\in[m+2,n]}$ are set to 0 if $m + 1 < n$.
  2. Choose $s \xleftarrow{\$} \mathbb{Z}_p$, then compute $k = e(g, g)^{\alpha s}$, $c_1 = g^s$ and $c_2 = \big(h \prod_{i=1}^{n} h_i^{y_i}\big)^s$, and finally output (k, ct).
– $\mathsf{Dec}(sk_{id}, id, ct, R)$: Given a private key sk_{id} for an identity id and a ciphertext $ct = (c_1, c_2)$ under the revocation set R, this algorithm returns $\perp$ if $id \in R$. Otherwise, it recovers the encapsulated key k by conducting the following steps:
  1. Define the vectors $\boldsymbol{x}_{id} = (x_1, \ldots, x_n)$ and $\boldsymbol{y}_R = (y_1, \ldots, y_n)$ as before.
  2. Compute $\hat{k} = k_1 \prod_{i=2}^{n} k_i^{y_i}$ and then return

$$k' = e\big(c_1, d_1 \cdot \hat{k}^{\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}}\big) \cdot e\big(c_2, d_2^{-\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}}\big).$$

The regular correctness follows readily from the IBR scheme [4]. For completeness, it is analyzed in details as follows. First, we know from the definitions of $\boldsymbol{x}_{id}$ and $\boldsymbol{y}_R$ that $\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle \neq 0$ *iff* $id \notin R$. Then we observe that

$$\hat{k} = h^r \prod_{i=2}^{n} \big(h_1^{-x_i y_i} \cdot h_i^{y_i}\big)^r = \big(h_1^{-\sum_{i=2}^{n} x_i y_i} \cdot (h \prod_{i=2}^{n} h_i^{y_i})\big)^r = \big(h_1^{-\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle} \cdot (h \prod_{i=1}^{n} h_i^{y_i})\big)^r,$$

so when $id \notin R$ we have that

$$\begin{aligned} k' &= e\big(c_1, d_1\big) \cdot \left(\frac{e(c_1, \hat{k})}{e(c_2, d_2)}\right)^{\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}} \\ &= e\big(g^s, g^{\alpha} h_1^r\big) \cdot \left(\frac{e\Big(g, h_1^{-\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle} \cdot h \prod_{i=1}^{n} h_i^{y_i}\Big)^{rs}}{e\Big(h \prod_{i=1}^{n} h_i^{y_i}, g\Big)^{rs}}\right)^{\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}} \\ &= e(g, g)^{\alpha s}. \end{aligned}$$

With regard to the extended correctness, it can be verified similarly. More specifically, we let $(msk', mpk') = (\alpha', e(g, g)^{\alpha'})$ be another master secret and public key pair, and $sk_{id'} = (d_1', d_2', k_1', \ldots, k_n') \leftarrow \mathsf{KeyExt}(msk', id')$ be a private key for identity id', such that

$$d_1' = g^{\alpha'} h_1^{r'},\ d_2' = g^{r'},\ k_1' = h^{r'},\ k_i' = \big(h_1^{-x_i'} h_i\big)^{r'} \text{ for } \forall i \in [2, n],$$

where $r' \xleftarrow{\$} \mathbb{Z}_p$ and $x_i' = id'^{i-1}$. Then it is easy to get via the above analysis that

$$\mathsf{Dec}(sk_{id'}, id', ct, R) = e\big(c_1, d_1' \cdot \hat{k}^{\frac{1}{\langle \boldsymbol{x}_{id'}, \boldsymbol{y}_R\rangle}}\big) \cdot e\big(c_2, d_2'^{-\frac{1}{\langle \boldsymbol{x}_{id'}, \boldsymbol{y}_R\rangle}}\big) = e(g, g)^{\alpha' s}$$

conditioned on $id' \notin R$, where $\boldsymbol{x}_{id'} = (x'_1, \ldots, x'_n)$ and $\hat{k} = k'_1 \prod_{i=2}^{n} k_i'^{y_i}$.

KEY-HOMOMORPHISM. The encapsulated key in this scheme is in the form of $f(msk, s) = e(g, g)^{msk \cdot s}$, where s is the random coin consumed in the encryption algorithm. Next, we show for any identity $id \in \mathbb{Z}_p$, master secret keys $\alpha, \alpha' \in \mathbb{Z}_p$ and randomness $r, r', s \in \mathbb{Z}_p$ that the key-homomorphic properties with respect to $\mathsf{KeyExt}(\cdot)$ and $f(\cdot)$ hold:

1. From the description above, we get that

$$\begin{aligned}
&\mathsf{KeyExt}(\alpha, id; r) \otimes \mathsf{KeyExt}(\alpha', id; r') \\
&= \big(g^{\alpha} h_1^{r}, g^{r}, h^{r}, (h_1^{-x_2} \cdot h_2)^{r}, \ldots, (h_1^{-x_n} \cdot h_n)^{r}\big) \otimes \\
&\quad \big(g^{\alpha'} h_1^{r'}, g^{r'}, h^{r'}, (h_1^{-x_2} h_2)^{r'}, \ldots, (h_1^{-x_n} h_n)^{r'}\big) \\
&= \big(g^{\alpha+\alpha'} h_1^{r+r'}, g^{r+r'}, h^{r+r'}, (h_1^{-x_2} h_2)^{r+r'}, \ldots, (h_1^{-x_n} h_n)^{r+r'}\big) \\
&= \mathsf{KeyExt}(\alpha + \alpha', id;\ r + r')
\end{aligned}$$

   where " $\otimes$ " over $\mathcal{SK} = \mathbb{G}^{n+1}$ is the coordinate-wise multiplication over $\mathbb{G}$.
2. As for $f(\cdot)$, it is clear that

$$f(\alpha, s) \odot f(\alpha', s) = e(g, g)^{\alpha s} \cdot e(g, g)^{\alpha' s} = e(g, g)^{(\alpha+\alpha')s} = f(\alpha + \alpha', s),$$

   where " $\odot$ " is the multiplication over $\mathbb{G}_T$.

SECURITY. The IRKEM scheme above is IND-sRL-CPA secure under the n-DBDHE assumption. This follows readily from the Embedding Lemma (see Proposition 1 in [3]) and the proof of the NIPE scheme in [38].

Now following the proposed generic construction in Section 3, we get the first PKEM scheme that features both unbounded punctures and constant-size ciphertexts, but subject to a bounded-number of tags per ciphertext.

4.2 KH-IRKEM with Compact Private Keys

The second IRKEM scheme is based on another NIPE scheme in [38] and proven secure under the DBDH assumption, which unlike the previous one is one of the weakest bilinear assumptions.

DBDH ASSUMPTION. Let $(\mathbb{G}, \mathbb{G}_T)$ be cyclic groups of prime order p with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The Decision Bilinear Diffie-Hellman (DBDH) problem is, given $\big(g, g^a, g^b, g^c, T\big)$ where $a, b, c \xleftarrow{\$} \mathbb{Z}_p, g \xleftarrow{\$} \mathbb{G}$ and $T \in \mathbb{G}_T$, to decide if $T = e(g, h)^{abc}$ or if T is a random element in $\mathbb{G}_T$.

DESCRIPTION. As before, this scheme consists of five polynomial-time algorithms $(\mathsf{Params}, \mathsf{MKGen}, \mathsf{KeyExt}, \mathsf{Enc}, \mathsf{Dec})$ as below:

- $\mathsf{Params}(1^\lambda, n)$: This algorithm takes a security parameter λ and an integer $n \in \mathbb{N}$, and generates a pair of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with bilinear map e and generator $g \xleftarrow{\$} \mathbb{G}$. Then it randomly chooses $\beta, b_1, \ldots, b_n \in \mathbb{Z}_p$, and computes $h = g^\beta$ and $h_i = g^{b_i}$ for all $i \in [1, n]$. Finally, it outputs the public parameters

$$pp = \big((\mathbb{G}, \mathbb{G}_T, e), g, h, \{h_i\}_{i \in [1,n]}\big).$$

– $\mathsf{MKGen}(pp)$: Given the public parameters pp, it randomly chooses $\alpha \in \mathbb{Z}_p$ and outputs the master secret key and master public key pair

$$(msk, mpk) = \big(\alpha, e(g,g)^{\alpha}\big).$$

– $\mathsf{KeyExt}(msk, id)$: Given a master secret key $msk = \alpha$ and an identity $id \in \mathbb{Z}_p$, the algorithm first defines a vector $\boldsymbol{x}_{id} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ such that $x_i = id^{i-1}$ for all $i \in [1, n]$, then it chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the private key $sk_{id} = (k_0, k_1, k_2) \in \mathbb{G}^3$ as

$$k_0 = g^{\alpha} h^r,\ k_1 = \big(\prod_{i=1}^{n} h_i^{x_i}\big)^r,\ k_2 = g^r.$$

– $\mathsf{Enc}(mpk, R)$: Given mpk and a revocation list $R = \{id_1, id_2, \ldots, id_m\}$ such that $m < n$, the algorithm generates the encapsulated key $k \in \mathbb{G}_T$ and ciphertext $ct = \big(c_0, \{c_{i,1}\}_{i\in[1,n]}\big) \in \mathbb{G}^{n+1}$ as:
  1. Define a vector $\boldsymbol{y}_R = (y_1, \ldots, y_n)$, where $\{y_i\}_{i\in[1,m+1]}$ are the coefficients of the polynomial $f_R(z) = \prod_{id_j \in R}(z - id_j) = \sum_{i=1}^{m+1} y_i \cdot z^{i-1}$, and all other coordinates $\{y_i\}_{i\in[m+2,n]}$ are set to 0 if $m + 1 < n$.
  2. Choose $s \xleftarrow{\$} \mathbb{Z}_p$, then compute $k = e(g,g)^{\alpha s}$, $c_0 = g^s$ and $c_{i,1} = (h^{y_i} h_i)^s$ for $\forall i \in [1, n]$, and finally output (k, ct).

– $\mathsf{Dec}(sk_{id}, id, ct, R)$: Given sk_{id} associated with id and $ct = \big(c_0, \{c_{i,1}\}_{i\in[1,n]}\big)$ associated with revocation set R, the algorithm returns $\perp$ if $id \in R$. Otherwise, it recovers the encapsulated key as follows:
  1. Define the vectors $\boldsymbol{x}_{id} = (x_1, \ldots, x_n)$ and $\boldsymbol{y}_R = (y_1, \ldots, y_n)$ as before.
  2. Compute $c_1 = \prod_{i=1}^{n} c_{i,1}^{x_i}$ and then return

$$k' = e\big(c_0, k_0 \cdot k_1^{\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}}\big) \cdot e\big(c_1, k_2^{-\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}}\big).$$

The regular correctness is verified as follows. First, we know from the definitions of $\boldsymbol{x}_{id}$ and $\boldsymbol{y}_R$ that $\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle \neq 0$ *iff* $id \notin R$. Then we observe that

$$c_1 = \prod_{i=1}^{n} c_{i,1}^{x_i} = \prod_{i=1}^{n} \big(h^{x_i y_i} \cdot h_i^{x_i}\big)^s = \big(h^{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle} \cdot \prod_{i=1}^{n} h_i^{x_i}\big)^s,$$

so when $id \notin R$ we have that

$$\begin{aligned} k' &= e\big(c_0, k_0\big) \cdot \Big(\frac{e(c_0, k_1)}{e(c_1, k_2)}\Big)^{\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}} \\ &= e\big(g^s, g^{\alpha} h^r\big) \cdot \left(\frac{e\big(g, \prod_{i=1}^{n} h_i^{x_i}\big)^{rs}}{e\big(h^{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle} \cdot \prod_{i=1}^{n} h_i^{x_i}, g\big)^{rs}}\right)^{\frac{1}{\langle \boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle}} \\ &= e(g,g)^{\alpha s}. \end{aligned}$$

As for the extended correctness, it can be validated in a similar way, as analyzed for the first construction. Here, we omit the details.

KEY-HOMOMORPHISM. In this scheme, the encapsulated key is $f(msk, s) = e(g,g)^{msk \cdot s}$ as well, where s is the encryption randomness. The group operations over e.g., $\mathcal{MSK}, \mathcal{SK}$ and $\mathcal{K}$ are defined as before. It is clear to see for all identity $id \in \mathbb{Z}_p$, any master secret keys $\alpha, \alpha' \in \mathbb{Z}_p$ and randomness $r, r', s \in \mathbb{Z}_p$, both the key-homomorphic properties with respect to $\mathsf{KeyExt}(\cdot)$ and $f(\cdot)$ hold:

1. $\mathsf{KeyExt}(\alpha, id; r) \otimes \mathsf{KeyExt}(\alpha', id; r') = \mathsf{KeyExt}(\alpha + \alpha', id;\ r + r')$
2. $f(\alpha, s) \odot f(\alpha', s) = f(\alpha + \alpha', s)$.

SECURITY. The IRKEM scheme above is IND-sRL-CPA secure under the DBDH assumption. This can be easily shown by following the proof of [38] and the Embedding Lemma (cf. Proposition 1 in [3]).

Then by applying the conversion in Section 3, we obtain a PKEM scheme with "compact" secret keys. Compared to the scheme [26] under the same assumption, the communication cost is comparable, but the computation cost is better on average, especially when the number of tags encrypted is large, e.g., n. In that case, the number of exponentiation over $\mathbb{G}$ in our encryption is $\mathcal{O}(n)$ rather than $\mathcal{O}(n^2)$ as [26], and the number of pairings in decryption is d-less than [26], where d is the number of punctures corresponding to the decryption key.

4.3 KH-IRKEM Supporting Unbounded Users

Next we give the third IRKEM scheme, in which the number of users per ciphertext is unbounded compared to the previous ones. This scheme is derived from the IBR scheme in [30] and proven secure under the q-MEBDH assumption.

q-MEBDH ASSUMPTION. Let $(\mathbb{G}, \mathbb{G}_T)$ be cyclic groups of prime order p with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The q-decisional Multi-Exponent Bilinear Diffie-Hellman (q-MEBDH) problem is, given

$$\begin{array}{cc} & T,\ g,\ g^s,\ e(g,g)^{\alpha} \\ \forall 1 \le i, j \le q & g^{a_i},\ g^{a_i s},\ g^{a_i a_j},\ g^{\alpha / a_i^2} \\ \forall 1 \le i, j, k \le q, i \ne j & g^{a_i a_j s},\ g^{\alpha a_j / a_i^2},\ g^{\alpha a_i a_j / a_k^2},\ g^{\alpha a_i^2 / a_j^2} \end{array}$$

where $\alpha, s, a_1, \ldots, a_q \xleftarrow{\$} \mathbb{Z}_p, g \xleftarrow{\$} \mathbb{G}$ and $T \in \mathbb{G}_T$, to decide if $T = e(g,g)^{\alpha s}$ or if T is a random element from $\mathbb{G}_T$.

DESCRIPTION. This scheme consists of five polynomial-time algorithms (Params, $\mathsf{MKGen}, \mathsf{KeyExt}, \mathsf{Enc}, \mathsf{Dec}$), which are described as follows:

– $\mathsf{Params}(1^\lambda)$: The algorithm takes a security parameter λ, and generates a tuple of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with bilinear map e. Then it randomly chooses $b \in \mathbb{Z}_p$ and $g, h \in \mathbb{G}$, and computes $g_1 = g^b, g_2 = g^{b^2}$ and $h_1 = h^b$. At last, it outputs the public parameters

$$pp = \big((\mathbb{G}, \mathbb{G}_T, e), g, g_1, g_2, h, h_1\big).$$

– $\mathsf{MKGen}(pp)$: Given the public parameters pp, it randomly chooses $\alpha \in \mathbb{G}$ and outputs the master secret key and public key pair

$$(msk, mpk) = \big(\alpha, e(g,g)^{\alpha}\big).$$

– $\mathsf{KeyExt}(msk, id)$: Given a master secret key $msk = \alpha$ and an identity $id \in \mathbb{Z}_p$, it chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the private key $sk_{id} = (k_0, k_1, k_2) \in \mathbb{G}^3$ as

$$k_0 = g^{\alpha} g_2^{r},\ k_1 = (g_1^{id} h)^{r},\ k_2 = g^{-r}.$$

– $\mathsf{Enc}(mpk, R)$: Given mpk and a revocation list $R = \{id_1, id_2, \ldots, id_m\}$, the algorithm selects $s, s_1, \ldots, s_m \xleftarrow{\$} \mathbb{Z}_p$ such that $s = \sum_{i=1}^{m} s_i$, and generates the encapsulated key $k \in \mathbb{G}_T$ and ciphertext $ct = \big(c_0, \{c_{i,1}, c_{i,2}\}_{i\in[1,m]}\big) \in \mathbb{G}^{2m+1}$ as follows:

$$k = e(g,g)^{\alpha s},\ c_0 = g^{s},\ c_{i,1} = g_1^{s_i},\ c_{i,2} = (g_2^{id_i} h_1)^{s_i} \text{ for } \forall i \in [1,m].$$

– $\mathsf{Dec}(sk_{id}, id, ct, R)$: Given a private key sk_{id} associated with id and a ciphertext $ct = \big(c_0, \{c_{i,1}, c_{i,2}\}_{i\in[1,m]}\big)$ under the revoked set $R = \{id_1, id_2, \ldots, id_m\}$, this algorithm returns $\perp$ if $id \in R$. Otherwise, it recovers the encapsulated key by computing:

$$k' = \frac{e(c_0, k_0)}{e(\prod_{i=1}^{m} c_{i,1}^{1/(id-id_i)}, k_1) \cdot e(\prod_{i=1}^{m} c_{i,2}^{1/(id-id_i)}, k_2)}$$

The regular correctness can be verified as follows. In the case of $id \notin R$, we have that

$$\begin{aligned}
& e(\prod_{i=1}^{m} c_{i,1}^{1/(id-id_i)}, k_1) \cdot e(\prod_{i=1}^{m} c_{i,2}^{1/(id-id_i)}, k_2) \\
=& \prod_{i=1}^{m} \big(e(c_{i,1}, k_1) \cdot e(c_{i,2}, k_2)\big)^{1/(id-id_i)} \\
=& \prod_{i=1}^{m} \big(e(g^{s_i}, (g_2^{id} h_1)^{r}) \cdot e((g_2^{id_i} h_1)^{s_i}, g^{-r})\big)^{1/(id-id_i)} \\
=& \prod_{i=1}^{m} e(g, g_2)^{s_i r},
\end{aligned}$$

and then we get that

$$\begin{aligned}
k' &= e(g^{s}, g^{\alpha} g_2^{r}) / \prod_{i=1}^{m} e(g, g_2)^{s_i r} \\
&= e(g,g)^{\alpha s} \cdot e(g, g_2)^{sr} / e(g, g_2)^{r \cdot \sum_{i=1}^{m} s_i} \\
&= e(g,g)^{\alpha s}.
\end{aligned}$$

Regarding the extended correctness, it can be analyzed as previous constructions, so we omit the details here.

Key-Homomorphism. In this scheme, the encapsulated key is the same as before, i.e., $f(msk, s) = e(g,g)^{msk \cdot s}$. The group operations are also defined similarly, e.g., "$\otimes$" over $\mathcal{SK} = \mathbb{G}^3$ is the coordinate-wise multiplication over $\mathbb{G}$ and

$\odot$ over $\mathcal{K}$ is the multiplication over $\mathbb{G}_T$. It is easy to verify that for any identity $id \in \mathbb{Z}_p$, master secret keys $\alpha, \alpha' \in \mathbb{Z}_p$ and randomness $r, r', s \in \mathbb{Z}_p$ the following properties hold:

1. $\mathsf{KeyExt}(\alpha, id; r) \otimes \mathsf{KeyExt}(\alpha', id; r') = \mathsf{KeyExt}(\alpha + \alpha', id;\ r + r')$,
2. $f(\alpha, s) \odot f(\alpha', s) = f(\alpha + \alpha', s)$.

Security. The IRKEM scheme presented above is IND-sRL-CPA secure under the q-MEBDH assumption. This holds straightforwardly, as this scheme is simply the key encapsulation version of the IBR scheme of [30].

By combining this scheme with our generic construction in Section 3, we obtain the first PKEM scheme that enjoys both compact master public key and "compact" punctured secret key. Here, the compactness of the latter means the key size depends only on the number of punctures, independent of the size of revoked set. Moreover, the number of revoked tags per ciphertext is unbounded in this scheme.

4.4 KH-IRKEM under DLIN Assumption

Finally, we present another IRKEM scheme featuring compact ciphertexts, which is derived from the NIPE scheme of [13] (cf. Section A.2). In contrast to the construction shown in Section 4.1, it is proven adaptively secure under the *standard* DLIN assumption.

DLIN Assumption. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be cyclic groups of prime order p with a non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The Decisional Linear (DLIN) problem is to distinguish between the distributions $\left(g^{x_1}, g^{x_2}, g^{x_1 y_1}, g^{x_2 y_2}, h^{y_1+y_2}\right)$ and $\left(g^{x_1}, g^{x_2}, g^{x_1 y_1}, g^{x_2 y_2}, h^{z}\right)$, where $x_1, x_2, y_1, y_2, z \xleftarrow{\$} \mathbb{Z}_p, g \xleftarrow{\$} \mathbb{G}_1$ and $h \xleftarrow{\$} \mathbb{G}_2$.

Description. Similarly, this scheme is composed of five polynomial-time algorithms (Params, MKGen, KeyExt, Enc, Dec):

– $\mathsf{Params}(1^\lambda, n)$: It takes as input a security parameter λ and an integer $n \in \mathbb{N}$, and generates cyclic groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ endowed with a bilinear map e. Then it samples $g \in \mathbb{G}_1, h \in \mathbb{G}_2, a_i, b_i \in \mathbb{Z}_p$ for $i \in \{1, 2\}$, and sets

$$\mathbf{A} = \begin{pmatrix} a_1 & \\ & a_2 \\ 1 & 1 \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} b_1 & \\ & b_2 \\ 1 & 1 \end{pmatrix}.$$

After that, it chooses $\mathbf{W}_1, \ldots, \mathbf{W}_n \xleftarrow{\$} \mathbb{Z}_p^{3\times 3}$ and outputs public parameters

$$pp = \left((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g, g^{\mathbf{A}}, \{g^{\mathbf{W}_i^\top \mathbf{A}}\}_{i\in[1,n]}, h, h^{\mathbf{B}}, \{h^{\mathbf{W}_i \mathbf{B}}\}_{i\in[1,n]}\right).$$

– $\mathsf{MKGen}(pp)$: Given the public parameters pp, it chooses $\boldsymbol{k} \xleftarrow{\$} \mathbb{Z}_p^3$, and outputs the master secret key and public key pair

$$(msk, mpk) = \left(\boldsymbol{k}, e(g, h)^{\mathbf{A}^\top \boldsymbol{k}}\right).$$

– $\mathsf{KeyExt}(msk, id)$: Given a master secret key $msk = \boldsymbol{k}$ and an identity $id \in \mathbb{Z}_p$, the algorithm defines a vector $\boldsymbol{x}_{id} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ such that $x_i = id^{i-1}$ for all $i \in [1, n]$, chooses $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^2$ and outputs the private key $sk_{id} = (\boldsymbol{k}_1, \{\boldsymbol{k}_{2,i}\}_{i\in[1,n]}) \in (\mathbb{G}_2^3)^{n+1}$ as

$$\boldsymbol{k}_1 = h^{\mathbf{B}\boldsymbol{r}},\ \boldsymbol{k}_{2,i} = h^{x_i\cdot\boldsymbol{k}+\mathbf{W}_i\mathbf{B}\boldsymbol{r}} \text{ for } \forall i \in [1, n].$$

– $\mathsf{Enc}(mpk, R)$: Given a master public key mpk and a revocation list $R = \{id_1, id_2, \ldots, id_m\}$ such that $m < n$, the algorithm produces the encapsulated key $k \in \mathbb{G}_T$ and ciphertext $ct = (\boldsymbol{c}_1, \boldsymbol{c}_2) \in (\mathbb{G}_1^3)^2$ as:
  1. Define a vector $\boldsymbol{y}_R = (y_1, \ldots, y_n)$, where $\{y_i\}_{i\in[1,m+1]}$ are the coefficients of the polynomial $f_R(z) = \prod_{id_j\in R}(z - id_j) = \sum_{i=1}^{m+1} y_i \cdot z^{i-1}$, and all other coordinates $\{y_i\}_{i\in[m+2,n]}$ are set to 0 if $m + 1 < n$.
  2. Select $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^2$ and compute

$$k = e(g, h)^{\boldsymbol{s}^\top\mathbf{A}^\top\boldsymbol{k}},\ \boldsymbol{c}_1 = g^{\mathbf{A}\boldsymbol{s}} \text{ and } \boldsymbol{c}_2 = g^{(\sum_{i=1}^n y_i\cdot\mathbf{W}_i^\top)\mathbf{A}\boldsymbol{s}}.$$

– $\mathsf{Dec}(sk_{id}, id, ct, R)$: Given a private key sk_{id} for id and a ciphertext $ct = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ under revocation list R, the algorithm returns $\perp$ if $id \in R$. Otherwise, it recovers the encapsulated key k as follows:
  1. Define the vectors $\boldsymbol{x}_{id} = (x_1, \ldots, x_n)$ and $\boldsymbol{y}_R = (y_1, \ldots, y_n)$ as before.
  2. Compute $\boldsymbol{k}_2 = \prod_{i=1}^n \boldsymbol{k}_{2,i}^{y_i}$ and then output

$$k' = \big(e(\boldsymbol{c}_1, \boldsymbol{k}_2)/e(\boldsymbol{c}_2, \boldsymbol{k}_1)\big)^{\frac{1}{\langle\boldsymbol{x}_{id},\boldsymbol{y}_R\rangle}}.$$

The regular correctness follows readily from the NIPE scheme of [13]. For completeness, we present the details below. As analyzed before, it holds that $\langle\boldsymbol{x}_{id}, \boldsymbol{y}_R\rangle \neq 0$ *iff* $id \notin R$. Further, we have that

$$\boldsymbol{k}_2 = \prod_{i=1}^n \big(h^{x_i\cdot\boldsymbol{k}+\mathbf{W}_i\mathbf{B}\boldsymbol{r}}\big)^{y_i} = h^{(\sum_{i=1}^n x_i y_i)\cdot\boldsymbol{k}+(\sum_{i=1}^n y_i\cdot\mathbf{W}_i)\mathbf{B}\boldsymbol{r}},$$

so when $id \notin R$ we get that

$$\begin{aligned}
k' &= \left(\frac{e\big(g^{\mathbf{A}\boldsymbol{s}},\ h^{(\sum_{i=1}^n x_i y_i)\cdot\boldsymbol{k}+(\sum_{i=1}^n y_i\cdot\mathbf{W}_i)\mathbf{B}\boldsymbol{r}}\big)}{e\big(g^{(\sum_{i=1}^n y_i\cdot\mathbf{W}_i^\top)\mathbf{A}\boldsymbol{s}},\ h^{\mathbf{B}\boldsymbol{r}}\big)}\right)^{\frac{1}{\langle\boldsymbol{x}_{id},\boldsymbol{y}_R\rangle}} \\
&= \left(\frac{e\big(g^{\mathbf{A}\boldsymbol{s}},\ h^{(\sum_{i=1}^n x_i y_i)\cdot\boldsymbol{k}}\big)e\big(g^{\mathbf{A}\boldsymbol{s}},\ h^{(\sum_{i=1}^n y_i\cdot\mathbf{W}_i)\mathbf{B}\boldsymbol{r}}\big)}{e\big(g^{(\sum_{i=1}^n y_i\cdot\mathbf{W}_i^\top)\mathbf{A}\boldsymbol{s}},\ h^{\mathbf{B}\boldsymbol{r}}\big)}\right)^{\frac{1}{\langle\boldsymbol{x}_{id},\boldsymbol{y}_R\rangle}} \\
&= e\left(g^{\mathbf{A}\boldsymbol{s}}, h^{\langle\boldsymbol{x}_{id},\boldsymbol{y}_R\rangle\cdot\boldsymbol{k}}\right)^{\frac{1}{\langle\boldsymbol{x}_{id},\boldsymbol{y}_R\rangle}} = e(g, h)^{\boldsymbol{s}^\top\mathbf{A}^\top\boldsymbol{k}}.
\end{aligned}$$

As for the extended correctness, it can be verified as before. In particular, we let $(msk', mpk') = (\boldsymbol{k}', e(g, h)^{\mathbf{A}^\top\boldsymbol{k}'})$ be another master key pair, and $sk_{id'} = (\boldsymbol{k}'_1, \{\boldsymbol{k}'_{2,i}\}_{i\in[1,n]}) \leftarrow \mathsf{KeyExt}(msk', id')$ be a private key for id', such that

$$\boldsymbol{k}'_1 = h^{\mathbf{B}\boldsymbol{r}'} \text{ and } \boldsymbol{k}'_{2,i} = h^{x'_i\cdot\boldsymbol{k}'+\mathbf{W}_i\mathbf{B}\boldsymbol{r}'} \text{ for } \forall i \in [1, n],$$

where $\boldsymbol{r}' \xleftarrow{\$} \mathbb{Z}_p^2$ and $x'_i = id'^{i-1}$. Then it is not difficult to obtain that

$$\mathsf{Dec}(sk_{id'}, id', ct, R) = \big(e(\boldsymbol{c}_1, \boldsymbol{k}'_2)/e(\boldsymbol{c}_2, \boldsymbol{k}'_1)\big)^{\frac{1}{\langle \boldsymbol{x}_{id'}, \boldsymbol{y}_R \rangle}} = e(g,h)^{\boldsymbol{s}^\top \mathbf{A}^\top \boldsymbol{k}'}$$

conditioned on $id' \notin R$, where $\boldsymbol{x}_{id'} = (x'_1, \ldots, x'_n)$ and $\boldsymbol{k}'_2 = \prod_{i=1}^n \boldsymbol{k}'^{\,y_i}_{2,i}$.

KEY-HOMOMORPHISM. In this scheme, the encapsulated key is in the form of $f(msk, \boldsymbol{s}) = e(g,h)^{\boldsymbol{s}^\top \mathbf{A}^\top \boldsymbol{k}}$, where $\boldsymbol{s}$ is the random coins of the encryption algorithm. Similar to previous analysis, it is not difficult to observe that, for any $id \in \mathbb{Z}_p$, master secret keys $\boldsymbol{k}, \boldsymbol{k}' \in \mathbb{Z}_p^3$ and randomness $\boldsymbol{r}, \boldsymbol{r}', \boldsymbol{s} \in \mathbb{Z}_p^2$, the following key-homomorphic properties with respect to $\mathsf{KeyExt}(\cdot)$ and $f(\cdot)$ hold:

1. $\mathsf{KeyExt}(\boldsymbol{k}, id; \boldsymbol{r}) \otimes \mathsf{KeyExt}(\boldsymbol{k}', id; \boldsymbol{r}') = \mathsf{KeyExt}(\boldsymbol{k} + \boldsymbol{k}', id;\ \boldsymbol{r} + \boldsymbol{r}')$,
2. $f(\boldsymbol{k}, \boldsymbol{s}) \odot f(\boldsymbol{k}', \boldsymbol{s}) = f(\boldsymbol{k} + \boldsymbol{k}', \boldsymbol{s})$.

SECURITY. This IRKEM scheme is IND-RL-CPA secure under the standard DLIN assumption. This can be argued by following the analysis of [13] and the Embedding Lemma (cf. Proposition 1 in [3]).

Then by plugging this scheme into our generic PKEM in Section 3, we obtain the first PKEM scheme with short ciphertext based on the standard assumption. Notice that, the other NIPE scheme (with short private key) in Section A.2 of [13] satisfies the desirable properties as well. Thus a new PKEM scheme with short secret key can be derived similarly, and we omit the details here.

5 Efficiency Comparison

In this part, we give a comprehensive comparison of our schemes with existing works [26,19,18], as shown in Table 1 and Table 2 (cf. Section 1). In the comparison, we use terms "exp" (resp. "exp$_T$") and "pair" to denote exponentiation in $\mathbb{G}$ (resp. $\mathbb{G}_T$) and bilinear pairing over $\mathbb{G}$, respectively. The column "unbounded punctures" (resp. ciphertext tags) in Table 1 refers to if unbounded punctures (resp. tags per ciphertext) is supported. For sake of simplicity, when comparing with the scheme of Green and Miers [26], we additionally add to their public key an element $e(g_1, g_2)$ used in their encryption, thus the pairing computation in encryption is replaced by an exponentiation in $\mathbb{G}_T$. In comparison with the basic (i.e., IBE-based) BFE scheme of [19] and the IBBE-based BFE scheme of [18], our schemes together with the scheme by Green and Miers support unbounded punctures and n tags per ciphertext for $n > 1$, rather than bounded punctures and unique tag in [19,18], but their scheme features fast puncture and decryption procedures. Moreover, our schemes do not suffer from non-negligible correctness errors and can be proven secure without random oracles. We note that, to compare with the generic IBBE-based BFE scheme in [18], we instantiate it with the efficient IBBE scheme featuring constant-size ciphertexts and private keys in [17]. In this work, our main concern is the PPE schemes with negligible correctness errors, so in the following the comparison is mainly conducted with Green and Miers's work [26].

Compared to the scheme of [26], our first scheme in Sec. 4.1 is based on a stronger assumption than [26]. It has a large size of punctured secret key (linear in the upper-bound n of tags per cipertext), but it features compact ciphertext of which the overhead consists of only 2 group elements in $\mathbb{G}$. Furthermore, it requires much fewer exponentiation evaluations for encryption, which is reduced from $O(\hat{n}n)$ to $O(\hat{n})$, and 33% less pairing computation for decryption. For the second scheme in Sec. 4.2, it also features compact[4] punctured secret key as [26]. Still, it is more efficient, especially for the case of encrypting messages under a large number of tags. Regarding the third scheme in Sec. 4.3, it enjoys the compact punctured secret key as well. In contrast, it also enjoys a short public key and allows unbounded tags per ciphertext. The disadvantage lies in that it relies on a stronger assumption q-MEBDH that appears less natural than q-DBDHE. Finally, the fourth scheme enjoys a comparable performance to the first one, but can achieve adaptive security under the standard DLIN assumption instead of a "q-type" one. More details of efficiency comparison are shown in Table 1 and Table 2.

We remark that here we focus on PPE itself and conduct no comparison of the PKEM schemes with the PFSKEM scheme in [28]. PFSKEM is inspired by the PFSE scheme in [26], which essentially combines the ideas of PPE and FS-PKE [11] for further reducing the decryption cost and punctured key size of PPE that grow linearly with the number of punctures. Similarly, we can also obtain PFSE schemes with distinct features based on the proposed PKEM schemes.

In addition, we note that the decryption complexity of our PKEM schemes is linear to the occurrence number of puncture operations (see Table 2). As argued in [26], however, it can be substantially mitigated like in [25] where the decryption of ABE is securely outsourced to a third party. Moreover, as the decryption of our construction is highly parallelized, it can be further optimized.

6 Further Discussion

Following the essential idea, our generic construction of PKEM can be optimized by further refining the correctness property of KH-IRKEM. To be more precise, we can further improve the computation efficiency of both decryption and puncture procedure by removing the use of distinguished tag t_0.

In particular, the secret key SK in the optimized version is the same as the master secret key msk of the underlying IRKEM, i.e., $SK = msk$ such that $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$. In the puncture procedure, the update on sk_0 (i.e., $sk'_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$) will be replaced by sequentially computing the remaining share of msk, i.e., $msk'_0 = msk_0 - msk_i$, in which case the punctured secret key for tags $\{t_\ell\}_{\ell=1}^{i}$ is in the form of $SK_i = \big(msk'_0, (sk_1, t_1), \ldots, (sk_i, t_i)\big)$ and msk_0 is the first component of SK_{i-1}. Note that in this case $SK_0 = msk$. For the decryption, the shared encapsulated key k_0 corresponding to tag t_0 is directly

[4] The compact here means the size of punctured secret key depends only on the number of punctures.

computed from the remaining share msk_0 of msk, instead of running the decryption algorithm of IRKEM (i.e., $k_0 \leftarrow \mathsf{IR.Dec}(sk_0, t_0, CT, T)$). To this goal, it is desired that a legally encapsulated key can be recovered correctly from msk_0 and CT along with the public parameters, as well. Therefore, the correctness property in Definition 7 needs to be further extended to include the following condition: there exists an efficiently computable key derivation function KDF, such that $\hat{k} = \mathsf{KDF}(pp, msk_1, ct_2)$. Fortunately, all our instantiations satisfy this additional property. In particular, the computation of $k_0 \leftarrow \mathsf{IR.Dec}(sk_0, t_0, CT, T)$ in the decryption will be replaced by $k_0 = \mathsf{KDF}(pp, msk_0, CT)$ and it is exactly $e(g, g^s)^{msk_0}$ in our instantiations, where g is part of pp, g^s is part of CT and s is the randomness of encryption.

As the optimized version do not change the asymptotic complexity of our PKEM schemes, we do not analyze its performance in the efficiency comparison.

7 Conclusion

We propose a generic method to construct public-key puncturable key encapsulation mechanism. Thus, we get the first modular way of designing the full-blown puncturable encryption with *negligible* correctness errors, by combining it with the standard decapsulation mechanism. To the end, we introduce a new concept of identity-based revocable encryption system, called key-homomorphic identity-based revocable key encapsulation mechanism with extended correctness. Furthermore, we present several instantiations of the new concept and obtains four concrete public-key puncturable encryption schemes with distinct features. Specifically, we get the first public-key puncturable encryption schemes with compact ciphertexts, and the first scheme allowing for both unbounded punctures and unbounded size of tag set in the ciphertext. We also get an efficient scheme based on the standard DBDH assumption that features both faster encryption and decryption when the size of tag set is large, compared to Green and Miers scheme based on the same assumption. Although we obtains some tradeoffs between distinct aspects in this work, it is still challenging to construct adaptively secure puncturable encryption scheme with e.g., both compact ciphertext and punctured keys in the standard model.

Acknowledgements The authors would like to thank the anonymous reviewers for their comments and suggestions. Also, they are grateful to Dr. Jie Chen for the helpful discussion. This work is supported in part by the Natural Science Foundation of China (No. 61802255), the project on Security Protection Technology of Embedded Components and Control Units in Power System Terminal (2019GW–12), the Data61-Monash Collaborative Research Project, and the Australian Research Council (ARC) Discovery Project (No. DP180102199).

References

1. Attrapadung, N.: Unbounded dynamic predicate compositions in attribute-based encryption. In: Advances in Cryptology - EUROCRYPT 2019 - 38th Annual Inter-

national Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I. pp. 34–67 (2019)
2. Attrapadung, N., Hanaoka, G., Ogawa, K., Ohtake, G., Watanabe, H., Yamada, S.: Attribute-based encryption for range attributes. In: Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings. pp. 42–61 (2016)
3. Attrapadung, N., Libert, B.: Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings. pp. 384–402 (2010)
4. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings. pp. 90–108 (2011)
5. Bellare, M., Cash, D., Miller, R.: Cryptography secure against related-key attacks and tampering. In: Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. pp. 486–503 (2011)
6. Bellare, M., Paterson, K.G., Thomson, S.: RKA security beyond the linear barrier: Ibe, encryption and signatures. In: Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. pp. 331–348 (2012)
7. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA. pp. 321–334 (2007)
8. Blazy, O., Kiltz, E., Pan, J.: (hierarchical) identity-based encryption from affine message authentication. In: Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. pp. 408–425 (2014)
9. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. pp. 213–229 (2001)
10. Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 1465–1482 (2017)
11. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. pp. 255–271 (2003)
12. Canetti, R., Raghuraman, S., Richelson, S., Vaikuntanathan, V.: Chosen-ciphertext secure fully homomorphic encryption. In: Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II. pp. 213–240 (2017)
13. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Advances in Cryptology - EUROCRYPT 2015 - 34th

Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. pp. 595–624 (2015)
14. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded ABE via bilinear entropy expansion, revisited. IACR Cryptology ePrint Archive **2018**, 116 (2018)
15. Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. SIAM J. Comput. **47**(6), 2157–2202 (2018)
16. D'Arco, P., Stinson, D.R.: On unconditionally secure robust distributed key distribution centers. In: Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings. pp. 346–363 (2002)
17. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings. pp. 200–215 (2007)
18. Derler, D., Gellert, K., Jager, T., Slamanig, D., Striecks, C.: Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange. IACR Cryptology ePrint Archive **2018**, 199 (2018)
19. Derler, D., Jager, T., Slamanig, D., Striecks, C.: Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange. In: Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III. pp. 425–455 (2018)
20. Derler, D., Krenn, S., Lorünser, T., Ramacher, S., Slamanig, D., Striecks, C.: Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In: Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I. pp. 219–250 (2018)
21. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. pp. 307–315 (1989)
22. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings. pp. 65–82 (2002)
23. Dodis, Y., Yampolskiy, A., Yung, M.: Threshold and proactive pseudo-random permutations. In: Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings. pp. 542–560 (2006)
24. Gong, J., Waters, B., Wee, H.: ABE for DFA from k-lin. In: Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. pp. 732–764 (2019)
25. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings (2011)
26. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015. pp. 305–320 (2015)
27. Günther, C.G.: An identity-based key-exchange protocol. In: Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings. pp. 29–37 (1989)

28. Günther, F., Hale, B., Jager, T., Lauer, S.: 0-rtt key exchange with full forward secrecy. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III. pp. 519–548 (2017)
29. Katsumata, S., Yamada, S.: Non-zero inner product encryption schemes from various assumptions: Lwe, ddh and dcr. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 158–188 (2019)
30. Lewko, A.B., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: 31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA. pp. 273–285 (2010)
31. MacKenzie, P.D., Reiter, M.K., Yang, K.: Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In: Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. pp. 171–190 (2004)
32. Martin, K.M., Safavi-Naini, R., Wang, H., Wild, P.R.: Distributing the encryption and decryption of a block cipher. Des. Codes Cryptography **36**(3), 263–287 (2005)
33. Mitzenmacher, M.: Bloom filters. In: Encyclopedia of Database Systems, Second Edition (2018)
34. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. pp. 349–366 (2012)
35. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. pp. 195–203 (2007)
36. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks (extended abstract). In: Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 19-21, 1991. pp. 51–59 (1991)
37. Sun, S., Yuan, X., Liu, J.K., Steinfeld, R., Sakzad, A., Vo, V., Nepal, S.: Practical backward-secure searchable encryption from symmetric puncturable encryption. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 763–780 (2018)
38. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: A framework and compact constructions for non-monotonic attribute-based encryption. In: Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings. pp. 275–292 (2014)

A Adaptive Security of Our PKEM

In this part, we will show the adaptive security of our PKEM. Briefly speaking, the proof is similar to that of Theorem 1 except that we will guess which puncture query is contained in the challenge query T^*. For completeness, the details are given as follows.

Theorem 3. *The proposed generic construction PKEM is IND-PUN-CPA secure, if the underlying IRKEM scheme is key-homomorphic and IND-RL-CPA secure. More precisely, for any PPT adversary $\mathcal{A}$ against the security of our PKEM scheme, it holds that*

$$\mathrm{Adv}^{\text{IND-PUN-CPA}}_{\mathit{PKEM},\mathcal{A}}(\lambda) \leq q \cdot \mathsf{Adv}^{\text{IND-RL-CPA}}_{\mathit{IRKEM},\mathcal{B}}(\lambda),$$

where q is the maximum number of puncture queries issued by $\mathcal{A}$ and $\mathcal{B}$ is some PPT algorithm against the security of the IRKEM scheme.

Proof. Similar to the proof of Theorem 1, it is conducted via a sequence of games. Henceforth, we use Win_i to denote the event that the adversary $\mathcal{A}$ wins in game G_i. For clarity, we assume that $\mathcal{A}$ makes at most q puncture queries, say $\{t_1, t_2, \ldots, t_q\}$, and at least one of them, say t_i for some $i \in [1, q]$, belongs to the set of challenge tags T^*. It is also assumed without loss of generality that the corrupt query is made after all q punctures. Then the current punctured secret key is sent back to $\mathcal{A}$ directly.

Game G_0: It is the real IND-PUN-CPA game, as described in Section 2.2. In more details, the challenger chooses a distinguished tag $t_0 \in \mathcal{T}$ and runs $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$, $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$ and $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk, t_0)$ to produce the public and secret key pair $(PK, SK) = \big((pp, mpk), (sk_0, t_0)\big)$. Then it returns to the adversary PK and uses SK to answer the puncture queries and challenge query as below. Also, it initializes an empty set $\mathtt{P}$ for keeping track of puncture queries.

- **Puncture**(t_i): The challenger chooses $msk_i \xleftarrow{\$} \mathcal{MSK}$, computes $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$, and updates the first component of SK_{i-1} as $sk_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$. It then sets $SK_i = \big((sk_0, t_0), \ldots, (sk_{i-1}, t_{i-1}), (sk_i, t_i)\big)$, where $SK_0 = SK$, and adds t_i to $\mathtt{P}$. Finally, it returns $SK_q = \big((sk_0, t_0), (sk_1, t_1), \ldots, (sk_q, t_q)\big)$ to $\mathcal{A}$ after all q puncture queries.
- **Challenge**: On input $T^* \subseteq \mathcal{T} \setminus \{t_0\}$, the challenger computes $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$ and randomly chooses $K_1^* \xleftarrow{\$} \mathcal{K}$. It then picks $b \xleftarrow{\$} \{0, 1\}$ and outputs (K_b^*, CT^*).

Eventually, $\mathcal{A}$ outputs a guess b', and wins the game if $b' = b$. From the security definition of PKEM (cf. Definition 2), we have that

$$\mathsf{Adv}^{\text{IND-PUN-CPA}}_{\mathsf{PKEM},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Win}_0] - \frac{1}{2} \right|.$$

Game G_1: This is identical to G_0 except the master secret keys $msk_1, msk_2, \ldots, msk_q \in \mathcal{MSK}$ are sampled beforehand instead of on-the-fly. In addition, they are used directly to answer the following puncture queries. Particularly, all queries are simulated as follows.

- **Puncture**(t_i): The challenger computes $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ and updates the first component of SK_{i-1} as $sk_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$,

by directly using msk_i chosen above. Then it sets $SK_i = \big((sk_0, t_0), \ldots, (sk_{i-1}, t_{i-1}), (sk_i, t_i)\big)$ and adds t_i to $\mathtt{P}$. Finally, it returns $SK_q = \big((sk_0, t_0), (sk_1, t_1), \ldots, (sk_q, t_q)\big)$ to $\mathcal{A}$ after all puncture queries.
- **Challenge**: On input $T^* \subseteq \mathcal{T} \setminus \{t_0\}$, the challenger computes $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$ and randomly picks $K_1^* \xleftarrow{\$} \mathcal{K}$. Then it outputs (K_b^*, CT^*) where $b \xleftarrow{\$} \{0, 1\}$.

At last, $\mathcal{A}$ outputs her guess b'. It can be seen that the modification introduced in this game does not affect the view of the adversary. Thus, we have that

$$\Pr[\mathsf{Win}_1] = \Pr[\mathsf{Win}_0].$$

Game G_2: The only difference of this game from G_1 is that the component sk_0 of the finally corrupted secret key SK_q is generated in an alternative way. Briefly, it is generated in a direct manner rather than by sequential updates (i.e., $sk_0 = sk_0 \otimes \mathsf{IR.KeyExt}(-msk_i, t_0)$). More specifically, the challenger runs $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$, and picks $msk_1, msk_2, \ldots, msk_q \in \mathcal{MSK}$ in advance. Then it sets $msk_0 = msk - \sum_{i=1}^{q} msk_i$ and computes $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk_0, t_0)$ for the distinguished tag $t_0 \in \mathcal{T}$. After that, it returns $PK = (pp, mpk)$ and simulates all queries with sk_0 and those master secret keys as below.

- **Puncture**(t_i): The challenger computes $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ with msk_i for the i-th puncture query t_i, and records t_i to $\mathtt{P}$. After receiving all q puncture queries from $\mathcal{A}$, it returns $SK_q = \big((sk_0, t_0), \ldots, (sk_q, t_q)\big)$. Recall that sk_0 is generated at the beginning.
- **Challenge**: After receiving the challenge T^* from $\mathcal{A}$, the challenger computes $(K_0^*, CT^*) \leftarrow \mathsf{IR.Enc}(mpk, T^*)$ and chooses $K_1^* \xleftarrow{\$} \mathcal{K}$. Then it chooses a random bit $b \in \{0, 1\}$ and returns (K_b^*, CT^*).

Finally, $\mathcal{A}$ outputs her guess b'. It is clear that the distribution of this game is identical to G_1, so we have

$$\Pr[\mathsf{Win}_2] = \Pr[\mathsf{Win}_1].$$

Game G_3: In this game, we randomly choose $j \in [1, q]$ and guess that the j-th puncture query t_j is contained in the set T^* of challenge tags. Recall that T^* contains at least one puncture query according to the security definition of PKEM (cf. Definition 2). Then the difference of this game from G_2 lies on the way of generating sk_0 and sk_j (associated with t_j).

In particular, the challenger in this game runs $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$, chooses $msk_0, \ldots, msk_{j-1}, msk_{j+1}, \ldots, msk_q \in \mathcal{MSK}$ uniformly at random, and sets $msk_j = msk - \sum_{i=0, \neq j}^{q} msk_i$. Then it uses msk_0 to compute $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk_0, t_0)$ for the distinguished tag $t_0 \in \mathcal{T}$ and uses msk_i to compute $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ for the i-th puncture query t_i, where $i \in [1, q]$. The challenge query is simulated in the same way as before.

It is not difficult to observe that the views of $\mathcal{A}$ in G_2 and G_3 are identical, since they essentially rely on the identical distributions $(msk, msk_0 = msk - \sum_{i=1}^{q} msk_i, msk_1, \ldots, msk_q)$ and $(msk, msk_0, \ldots, msk_{j-1}, msk_j = msk - \sum_{i=1,\neq j}^{q} msk_i, msk_{j+1}, \ldots, msk_q)$, respectively. Therefore, we get that

$$\Pr[\mathsf{Win}_3] = \Pr[\mathsf{Win}_2].$$

Next we proceed to show the advantage of $\mathcal{A}$ in G_3 is negligible in λ under the security of the the IRKEM scheme, which is formally stated in Lemma 2.

Lemma 2. *Provided the underlying IRKEM scheme Σ is IND-RL-CPA secure and key-homomorphic, then the advantage of $\mathcal{A}$ in G_3 is negligible in λ. That is,*

$$\left|\Pr[\mathsf{Win}_3] - \frac{1}{2}\right| \leq q \cdot \mathsf{Adv}^{\text{IND-RL-CPA}}_{\mathsf{IRKEM},\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ is some PPT algorithm against the security of the IRKEM scheme.

Proof (of Lemma 2). Suppose for contradiction that there exists an efficient adversary $\mathcal{A}$ winning in G_3 with non-negligible advantage, then we can construct an efficient algorithm $\mathcal{B}$ to break the IND-RL-CPA security of the underlying IRKEM scheme Σ, as follows.

After receiving from the challenger of IRKEM the public parameters pp and master public key mpk, such that $pp \leftarrow \mathsf{IR.Params}(1^\lambda, n)$ and $(mpk, msk) \leftarrow \mathsf{IR.MKGen}(pp)$, $\mathcal{B}(1^\lambda)$ sets $PK = (pp, mpk)$ and returns it to the adversary $\mathcal{A}$. Then $\mathcal{B}$ chooses uniformly at random $msk_0, \ldots, msk_{j-1}, msk_{j+1}, \ldots, msk_q$ from $\mathcal{MSK}$, and generates $sk_0 \leftarrow \mathsf{IR.KeyExt}(msk_0, t_0)$ for a distinguished tag $t_0 \in \mathcal{T}$ of his choice. After that, $\mathcal{B}$ simulates the puncture queries and challenge query as follows:

- **Puncture**(t_i): For the i-th puncture query $t_i \in \mathcal{T}$, $\mathcal{B}$ uses msk_i chosen initially to generate $sk_i \leftarrow \mathsf{IR.KeyExt}(msk_i, t_i)$ if $i \neq j$. Otherwise, $\mathcal{B}$ forwards t_j to the key extraction oracle of IRKEM and gets the corresponding private key sk'_j. Then $\mathcal{B}$ computes $sk_j = sk'_j \otimes \mathsf{IR.KeyExt}(-\sum_{i=0,\neq j}^{q} msk_i, t_j)$, and adds t_i to $\mathtt{P}$. At last, $\mathcal{B}$ returns $SK_q = \big((sk_0, t_0), (sk_1, t_1), \ldots, (sk_q, t_q)\big)$ to $\mathcal{A}$ after all puncture queries.
- **Challenge**: After receiving the challenge T^* from $\mathcal{A}$, $\mathcal{B}$ checks whether or not his guess is right (i.e., $t_j \in T^*$). If true, $\mathcal{B}$ forwards T^* to the challenger of IRKEM and sends to $\mathcal{A}$ the response (K^*_b, CT^*), then he returns what $\mathcal{A}$ outputs. Otherwise, $\mathcal{B}$ outputs a random guess.

Now we complete the description of algorithm $\mathcal{B}$. Next we continue to analyze the advantage of $\mathcal{B}$ against the underlying IRKEM scheme. In the following, we denote by Guess the event that $\mathcal{B}$ correctly guesses which puncture query is contained in $T*$. Then we get that

$$\begin{aligned}
&\Pr[\mathcal{B}(1^\lambda, \mathsf{View}) = b] - \tfrac{1}{2} \\
&= \Pr[\mathcal{B}(1^\lambda, \mathsf{View}) = b \wedge \mathsf{Guess}] + \Pr[\mathcal{B}(1^\lambda, \mathsf{View}) = b \wedge \overline{\mathsf{Guess}}] - \tfrac{1}{2} \\
&= \Pr[\mathsf{Win}_3] \cdot \Pr[\mathsf{Guess}] + \tfrac{1}{2}\Pr[\overline{\mathsf{Guess}}] - \tfrac{1}{2} \\
&= (\Pr[\mathsf{Win}_3] - \tfrac{1}{2}) \cdot \Pr[\mathsf{Guess}],
\end{aligned}$$

where View denotes the view of $\mathcal{B}$ in the IRKEM game, and the second equality follows from the perfect simulation of G_3 conditioned on the event Guess. Thus we have that

$$\mathsf{Adv}^{\text{IND-RL-CPA}}_{\mathsf{IRKEM},\mathcal{B}}(\lambda) = \left|\Pr[\mathcal{B}(1^\lambda, \mathsf{View}) = b] - \frac{1}{2}\right| \geq \frac{1}{q} \cdot \left|\Pr[\mathsf{Win}_3] - \frac{1}{2}\right|.$$

Putting all together, we get that

$$\mathsf{Adv}^{\text{IND-PUN-CPA}}_{\mathsf{PKEM},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Win}_3] - \frac{1}{2}\right| \leq q \cdot \mathsf{Adv}^{\text{IND-RL-CPA}}_{\mathsf{IRKEM},\mathcal{B}}(\lambda).$$