2020

Puncturable encryption: A generic construction from delegatable fully key-homomorphic encryption

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Dung Hoang Duong
University of Wollongong, hduong@uow.edu.au

Quoc Huy Le
University of Wollongong, qhl576@uowmail.edu.au

Josef Pieprzyk

Puncturable encryption: A generic construction from delegatable fully key-homomorphic encryption

Abstract

© Springer Nature Switzerland AG 2020. Puncturable encryption (PE), proposed by Green and Miers at IEEE S&P 2015, is a kind of public key encryption that allows recipients to revoke individual messages by repeatedly updating decryption keys without communicating with senders. PE is an essential tool for constructing many interesting applications, such as asynchronous messaging systems, forward-secret zero round-trip time protocols, public-key watermarking schemes and forward-secret proxy re-encryptions. This paper revisits PEs from the observation that the puncturing property can be implemented as efficiently computable functions. From this view, we propose a generic PE construction from the fully key-homomorphic encryption, augmented with a key delegation mechanism (DFKHE) from Boneh et al. at Eurocrypt 2014. We show that our PE construction enjoys the selective security under chosen plaintext attacks (that can be converted into the adaptive security with some efficiency loss) from that of DFKHE in the standard model. Basing on the framework, we obtain the first post-quantum secure PE instantiation that is based on the learning with errors problem, selective secure under chosen plaintext attacks (CPA) in the standard model. We also discuss about the ability of modification our framework to support the unbounded number of ciphertext tags inspired from the work of Brakerski and Vaikuntanathan at CRYPTO 2016.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Susilo, W., Duong, H., Le, Q. & Pieprzyk, J. (2020). Puncturable encryption: A generic construction from delegatable fully key-homomorphic encryption. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12309 LNCS 107-127.

Puncturable Encryption: A Generic Construction from Delegatable Fully Key-Homomorphic Encryption

Willy Susilo[1], Dung Hoang Duong[1], Huy Quoc Le[1,2], Josef Pieprzyk[2,3]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong NSW 2522, Australia.
{wsusilo,hduong}@uow.edu.au, qhl576@uowmail.edu.au
[2] CSIRO Data61, Sydney, NSW 2015, Australia,
and Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.
Josef.Pieprzyk@data61.csiro.au

Abstract. Puncturable encryption (PE), proposed by Green and Miers at IEEE S&P 2015, is a kind of public key encryption that allows recipients to revoke individual messages by repeatedly updating decryption keys without communicating with senders. PE is an essential tool for constructing many interesting applications, such as asynchronous messaging systems, forward-secret zero round-trip time protocols, public-key watermarking schemes and forward-secret proxy re-encryptions. This paper revisits PEs from the observation that the puncturing property can be implemented as efficiently computable functions. From this view, we propose a generic PE construction from the fully key-homomorphic encryption, augmented with a key delegation mechanism (DFKHE) from Boneh et al. at Eurocrypt 2014. We show that our PE construction enjoys the selective security under chosen plaintext attacks (that can be converted into the adaptive security with some efficiency loss) from that of DFKHE in the standard model. Basing on the framework, we obtain the first post-quantum secure PE instantiation that is based on the learning with errors problem, selective secure under chosen plaintext attacks (CPA) in the standard model. We also discuss about the ability of modification our framework to support the unbounded number of ciphertext tags inspired from the work of Brakerski and Vaikuntanathan at CRYPTO 2016.

Key words: Puncturable encryption, attribute-based encryption, learning with errors, arithmetic circuits, fully key-homomorphic encryption, key delegation

1 Introduction

Puncturable encryption (PE), proposed by Green and Miers [18] in 2015, is a kind of public key encryption, which can also be seen as a tag-based encryption (TBE), where both encryption and decryption are controlled by tags. Similarly to TBE, a plaintext in PE is encrypted together with tags, which are called *ciphertext tags*. In addition, the puncturing property of PE allows to produce new punctured secret keys associated some *punctures* (or *punctured tags*). Although the new keys (*puncture keys*) differ from the old ones, they still allow recipients to decrypt old ciphertexts as long as chosen *punctured tags* are different from tags embedded in the ciphertext. The puncturing property is very useful when the current decryption key is compromised. In a such situation, a recipient merely needs to update his key using the puncturing mechanism. PE is also useful when there is a need to revoke decryption capability from many users in order to protect some sensitive information (e.g., a time period or user identities). In this case, the puncturing mechanism is called for time periods or user identities.

Also, PE can provide forward security in a fine-grained level. Forward security, formulated in [19] in the context of key-exchange protocols, is a desired security property that helps to reduce a security risk caused by key exposure attacks. In particular, forward secure encryption (FSE)

guarantees confidentiality of old messages, when the current secret key has been compromised. Compared to PE, FSE provides a limited support for revocation of decryption capability. For instance, it is difficult for FSE to control decryption capability for any individual ciphertext (or all ciphertexts) produced during a certain time period, which, in contrast, can be easily done with PE.

Due to the aforementioned advantages, PE has become more and more popular and has been used in many important applications in such as asynchronous messaging transport systems [18], forward-secure zero round–trip time (0-RTT) key-exchange protocols [15, 20], public-key watermarking schemes [12] and forward-secure proxy re-encryptions [16].

Related Works. Green and Miers [18] propose the notion of PE and also present a specific ABE-based PE instantiation. The instantiation is based on the decisional bilinear Diffie-Hellman assumption (DBDH) in bilinear groups and is proven to be CPA secure in the random oracle model (ROM). Following the work [18], many other constructions have been proposed such as [10, 12, 15, 20, 26] (see Table 1 for a summary). For instance, Günther et al. [20] have provided a generic PE construction from *any* selectively secure hierarchical identity-based key encapsulation (HIBEKEM) combined with an *any* one time signature (OTS). In fact, the authors of [20] claim that their framework can be instantiated as the first post-quantum PE. Also, in the work [20], the authors present the first PE-based forward-secret zero round-trip time protocol with full forward secrecy. However, they instantiate PE that is secure in the standard model (SDM) by combining a (DDH)-based HIBE with a OTS based on discrete logarithm. The construction supports a predetermined number of ciphertext tags as well as a limited number of punctures. Derler et al. [15] introduce the notion of Bloom filter encryption (BFE), which can be converted to PE. They show how to instantiate BFE using identity-based encryption (IBE) with a specific construction that assumes intractability of the bilinear computational Diffie-Hellman (BCDH) problem. Later, Derler et. al. [14] extend the result of [15] and give a generic BFE construction from identity-based broadcast encryption (IBBE). The instantiation in [14] is based on a generalization of the Diffie-Hellman exponent (GDDHE) assumption in parings. However, the construction based on BFE suffers from *non-negligible correctness error*. This excludes it from applications that require negligible correctness error, as discussed in [26]. Most recently, Sun et al. [26] have introduced a new concept, which they call key-homomorphic identity-based revocable key encapsulation mechanism (KH-IRKEM) with extended correctness, from which they obtain a modular design of PE with *negligible correctness errors*. In particular, they describe four modular and compact instantiations of PE, which are secure in SDM. However, all of them are based on hard problems in pairings, namely q-decision bilinear Diffie-Hellman exponent problem (q–DBDHE), the decision bilinear Diffie-Hellman problem(DBDH), the q-decisional multi-exponent bilinear Diffie-Hellman (q-MEBDH) problem and the decisional linear problem (DLIN). We emphasize that all existing instantiations mentioned above are insecure against quantum adversaries. Some other works like [10,12] based PE on the notion of indistinguishability obfuscation, which is still impractical. The reader is referred to [26] for a state-of-the-art discussion.

To the best of our knowledge, there has been no specific lattice-based PE instantiation, which simultaneously enjoys negligible correctness error as well as post-quantum security in the standard model.

Our Contribution. We first give a *generic* construction of PE from *delegatable fully key-homomorphic encryption* (DFKHE) framework. The framework is a generalisation of fully key-

homomorphic encryption (FKHE) [5] by adding a key delegation mechanism. The framework is closely related to the functional encryption [7].

We also present an explicit PE construction based on lattices. Our design is obtained from LWE-based DFKHE that we build using FKHE for the learning with errors (LWE) setting [5]. This is combined with the key delegation ability supplied by the lattice trapdoor techniques [1, 11, 17]. Our lattice FE construction has the following characteristics:

- It supports a predetermined number of ciphertext tags per ciphertext. The ciphertext size is short and depends linearly on the number of ciphertext tags, which is fixed in advance. However, we note that following the work of Brakerski and Vaikuntanathan [8], our construction might be extended to obtain a variant that supports unbounded number of ciphertext tags (see Section 5 for a detailed discussion),
- It works for a predetermined number of punctures. The size of decryption keys (i.e., puncture keys) increases quadratically with the number of punctured tags,
- It offers selective CPA security in the standard model (that can be converted into full CPA security using the complexity leveraging technique as discussed in [9], [21] [4], [5]). This is due to CPA security of LWE-based underlying DFKHE (following the security proof for the generic framework).
- It enjoys post-quantum security and negligible correctness errors.

Table 1 compares our work with the results obtained by other authors. At first sight, the FE framework based on key homomorphic revocable identity-based (KH-IRKEM) [26] looks similar to ours. However, both frameworks are different. While key-homomorphism used by us means the capacity of transforming (as claimed in [5, Subsection 1.1]) "*an encryption under key* $\mathbf{x}$ *into an encryption under key* $f(\mathbf{x})$", key-homomorphism defined in [26, Definition 8] reflects the ability of preserving the algebraic structure of (mathematical) groups.

Overview and Techniques. We start with a high-level description of fully-key homomorphism encryption (FHKE), which was proposed by Boneh et al. [5]. Afterwards, we introduce what we call the *delegetable fully-key homomorphism encryption (DFHKE)*. At high-level description, FKHE possesses a mechanism that allows to convert a ciphertext $ct_{\mathbf{x}}$ (associated with a public variable $\mathbf{x}$) into the evaluated one ct_f for the same plaintext (associated with the pair (y, f)), where f is an efficiently computable function and $f(\mathbf{x}) = y$. In other words, FKHE requires a special key-homomorphic evaluation algorithm, called Eval, such that $ct_f \leftarrow \mathsf{Eval}(f, ct_{\mathbf{x}})$. In order to successfully decrypt an evaluated ciphertext, the decryptor needs to evaluate the initial secret sk to get sk_f. An extra algorithm, called KHom, is needed to do this, i.e. $sk_f \leftarrow \mathsf{KHom}(sk, (y, f))$. A drawback of FKHE is that it supports only a single function f.

Actually, we'd like to perform key-homomorphic evaluation for many functions $\{f_1, \cdots, f_k\}$ that belong to a family $\mathcal{F}$. To meet the requirement and obtain DFKHE, we generalise FKHE by endowing it with two algorithms $\mathsf{ExtEval}$ and KDel. The first algorithm transforms $(ct_{\mathbf{x}}, \mathbf{x})$ into $(ct_{f_1,\cdots,f_k}, (y, f_1, \cdots f_k))$, where $f_1(\mathbf{x}) = \cdots = f_k(\mathbf{x}) = y$. This is written as $ct_{f_1,\cdots,f_k} \leftarrow \mathsf{ExtEval}(f_1, \cdots, f_k, ct_{\mathbf{x}})$. The second algorithm allows to delegate the secret key step by step for the next function or $sk_{f_1,\cdots,f_k} \leftarrow \mathsf{KDel}(sk_{f_1,\cdots,f_{k-1}}, (y, f_k))$.

Our generic PE framework is inspired by a simple but subtle observation that puncturing property requires equality of ciphertext tags and punctures. This can be provided by functions that can be efficiently computed by arithmetic circuits. We call such functions *equality test functions*. Note that for PE, ciphertext tags play the role of variables $\mathbf{x}$'s and equality test functions act as functions f's defined in FKHE. For FE, one more puncture added defines one extra equality test function, which needs a delegation mechanism to take the function

Literature	From	Assumption	Security Model	#Tags	#Punctures	Post-quantum	Negl. Corr. Error
Green [18]	ABE	DBDH	ROM	$< \infty$	∞	×	✓
Günther [20]	any HIBE + any OTS	DDH (HIBE) + DLP (OTS)	SDM	$< \infty$	$< \infty$	×	✓
Derler [14]	BFE (IBBE)	GDDHE	**ROM***	1	$< \infty$	×	×
Derler [15]	BFE (IBE)	BCDH	ROM	1	$< \infty$	×	×
Sun [26]	KH-IRKEM	q–DBDHE	SDM	$< \infty$	∞	×	✓
		DBDH		$< \infty$	∞	×	
		q–MEBDH		∞	∞	×	
		DLIN		$< \infty$	∞	×	
This work	DFKHE	DLWE	SDM	$< \infty$	$< \infty$	✓	✓

Table 1: Comparison of some existing PE constructions in the literature with ours. Note that, here all works are being considered in the CPA security setting. The notation "$< \infty$" means "bounded" or "predetermined", while "∞" means "unlimited" or "arbitrary". The column entitled "Post-quantum" says whether the specific construction in each framework is post-quantum secure or not regardless its generic framework. The last column mentions to supporting the negligible correctness error. **ROM***: For the BFE-based FE basing on the IBBE instantiation of Derler et al. [14], we note that, the IBBE instantiation can be modified to remove ROM, as claimed by Delerablée in [13, Subection 3.2]

into account. We note that the requirement can be easily met using the same idea as the key delegation mentioned above. In order to be able to employ the idea of DFKHE for $(y_0, \mathcal{F})$ to PE, we define an efficiently computable family $\mathcal{F}$ of equality test functions $f_{t^*}(\mathbf{t})$ allowing us to compare the puncture t^* with ciphertext tags $\mathbf{t} = (t_1, \cdots, t_d)$ under the definition that $f_{t^*}(\mathbf{t}) = y_0$ iff $t^* \neq t_j \forall j \in [d]$, for some fixed value y_0.

For concrete DHKHE and PE constructions, we employ the LWE-based FKHE proposed in [5]. In this system, the ciphertext is $ct = (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_1, \cdots, \mathbf{c}_d, \mathbf{c}_{\mathsf{out}})$, where $\mathbf{c}_i = (t_i\mathbf{G} + \mathbf{B}_i)^T\mathbf{s} + \mathbf{e}_i$ for $i \in [d]$. Here the gadget matrix $\mathbf{G}$ is a special one, whose associated trapdoor $\mathbf{T}_{\mathbf{G}}$ (i.e., a short basis for the q-ary lattice $\Lambda_q^{\perp}(\mathbf{G})$) is publicly known (see [22] for details). Also, there exist three evaluation algorithms named $\mathsf{Eval}_{\mathsf{pk}}$, $\mathsf{Eval}_{\mathsf{ct}}$ and $\mathsf{Eval}_{\mathsf{sim}}$ [5], which help us to homomorphically evaluate a circuit (function) for a ciphertext ct. More specifically, from $\mathbf{c}_i := [t_i\mathbf{G} + \mathbf{B}_i]^T\mathbf{s} + \mathbf{e}_i$, where $\|\mathbf{e}_i\| < \delta$ for all $i \in [d]$, and a function $f : (\mathbb{Z}_q)^d \to \mathbb{Z}_q$, we get $\mathbf{c}_f = [f(t_1, \cdots, t_d)\mathbf{G} + \mathbf{B}_f]^T\mathbf{s} + \mathbf{e}_f, \|\mathbf{e}_f\| < \Delta$, where $\mathbf{B}_f \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f, (\mathbf{B}_i)_{i=1}^d)$, $\mathbf{c}_f \leftarrow \mathsf{Eval}_{\mathsf{ct}}(f, ((t_i, \mathbf{B}_i, \mathbf{c}_i))_{i=1}^d)$, and $\Delta < \delta \cdot \beta$ for some β sufficiently small. The algorithm $\mathsf{ExtEval}$ mentioned above can be implemented calling many times $\mathsf{Eval}_{\mathsf{pk}}$,$\mathsf{Eval}_{\mathsf{ct}}$, each time for each function. Meanwhile, $\mathsf{Eval}_{\mathsf{sim}}$ is only useful in the simulation for the security proof. In the LWE-based DFKHE construction, secret keys are trapdoors for q-ary lattices of form $\Lambda_q^{\perp}([\mathbf{A}|\mathbf{B}_{f_1}|\cdots|\mathbf{B}_{f_k}])$. For the key delegation KDel, we can utilize the trapdoor techniques [1, 11, 17] . For the LWE-based PE instantiation, we employ the equality test function with $y_0 := 0 \pmod q$. Namely, for a puncture t^* and a list of ciphertext tags $t_1, \cdots, t_d$ we define $f_{t^*}(t_1, \cdots, t_d) := eq_{t^*}(t_1) + \cdots + eq_{t^*}(t_d)$, where $eq_{t^*} : \mathbb{Z}_q \to \mathbb{Z}_q$ satisfying that $\forall t \in \mathbb{Z}_q$, $eq_{t^*}(t) = 1 \pmod q$ iff $t = t^*$, otherwise $eq_{t^*}(t) = 0 \pmod q$. Such functions has also been employed in [6] to construct a privately puncturable pseudorandom function. It follows from generic construction that our PE instantiation is selective CPA-secure.

Public key size	$O((d+1)\cdot n^2\log^2 q)$
Secret key size	$O(n^2\log^2 q\cdot\log(n\log q))$
Punctured key size	$(\eta+1)\cdot n\log q\cdot(O(\log(\beta_{\mathcal{F}})+\eta\cdot\log(n\log q)))$
Ciphertext size	$O((d+2)\cdot n\log^2 q))$

Table 2: Keys and ciphertext's size of our LWE-based PE as functions in number of ciphertext tags d and number of punctures η.

Efficiency. Table 2 summarizes the asymptotic bit-size of public key, secret key, punctured key and ciphertext. We can see that the public key size is a linear function in the number of ciphertext tags (i.e., d). The (initial) secret key size is independent of both d and η (the number of punctures). The punctured key (decryption key) size is a quadratic function of η. Lastly, the ciphertext size is a linear function of d.

On unbounded ciphertext tags. We believe that our framework can be extended to support unbounded number of ciphertext tags by exploiting the interesting technique of [8]. The key idea of [8] is to use homomorphic evaluation of a family pseudorandom functions. This helps to stretch a predetermined parameter (e.g., the length of a seed) to an arbitrary number of ciphertext tags. The predetermined parameter will be used to generate other public parameters (e.g., public matrices). More details is given in Section 5.

Paper Organization. In Section 2, we review some background related to this work. Our main contributions are presented in Section 3 and Section 4. We formally define DFKHE and the generic PE construction from DFKHE in Section 3. Section 4 is dedicated to the LWE-based instantiation of DFKHE and the induced PE. Section 5 discusses on the feasibility of transforming our proposed LWE-based PE to work well with unbounded ciphertext tags. This work is concluded in Section 6.

2 Preliminaries

2.1 Framework of Puncturable Encryption

Syntax of puncturable encryption. For a security parameter λ, let $d = d(\lambda)$, $\mathcal{M} = \mathcal{M}(\lambda)$ and $\mathcal{T} = \mathcal{T}(\lambda)$ be maximum number of tags per ciphertext, the space of plaintexts and the set of valid tags, respectively. Puncturable encryption (PE) is a collection of the following four algorithms KeyGen, $\mathsf{Encrypt}$, $\mathsf{Puncture}$ and $\mathsf{Decrypt}$:

- $(pk, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda, d)$: For a security parameter λ and the maximum number d of tags per ciphertext, the probabilistic polynomial time (PPT) algorithm KeyGen outputs a public key pk and an initial secret key sk_0.
- $ct \leftarrow \mathsf{Encrypt}(pk, \mu, \{t_1, \cdots, t_d\})$: For a public key pk, a message μ, and a list of tags $t_1, \cdots, t_d$, the PPT algorithm $\mathsf{Encrypt}$ returns a ciphertext ct.
- $sk_i \leftarrow \mathsf{Puncture}(pk, sk_{i-1}, t_i^*)$: For any $i > 1$, on input pk, sk_{i-1} and a tag t_i^*, the PPT algorithm $\mathsf{Puncture}$ outputs a punctured key sk_i that decrypts any ciphertexts, except for the ciphertext encrypted under any list of tags containing t_i^*.
- $\mu/\perp \leftarrow \mathsf{Decrypt}(pk, sk_i, (ct, \{t_1, \cdots, t_d\}))$: For input pk, a ciphertext ct, a secret key sk_i, and a list of tags $\{t_1, \cdots, t_d\}$, the deterministic polynomial time (DPT) algorithm $\mathsf{Decrypt}$ outputs either a message μ if the decryption succeeds or $\perp$ if it fails.

Correctness. The correctness requirement for PE is as follows:
For all $\lambda, d, \eta \geq 0, t_1^*, \cdots, t_\eta^*, t_1, \cdots, t_d \in \mathcal{T}$, $(pk, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda, d)$, $sk_i \leftarrow \mathsf{Punc}(pk, sk_{i-1}, t_i^*)$, $\forall i \in [\eta]$, $ct = \mathsf{Encrypt}(pk, \mu, \{t_1, \cdots, t_d\})$, we have

- If $\{t_1^*, \cdots, t_\eta^*\} \cap \{t_1, \cdots, t_d\} = \emptyset$, then $\forall i \in \{0, \cdots, \eta\}$,

$$\Pr[\mathsf{Decrypt}(pk, sk_i, (ct, \{t_1, \cdots, t_d\})) = \mu] \geq 1 - \mathsf{negl}(\lambda).$$

- If there exist $j \in [d]$ and $k \in [\eta]$ such that $t_k^* = t_j$, then $\forall i \in \{k, \cdots, \eta\}$,

$$\Pr[\mathsf{Decrypt}(pk, sk_i, (ct, \{t_1, \cdots, t_d\})) = \mu] \leq \mathsf{negl}(\lambda).$$

Definition 1 (Selective Security of PE). *PE is IND-sPUN-ATK if the advantage of any PPT adversary $\mathcal{A}$ in the game $\mathsf{IND\text{-}sPUN\text{-}ATK}_{\mathsf{PE}}^{\mathsf{sel},\mathcal{A}}$ is negligible, where $ATK \in \{CPA, CCA\}$. Formally,*

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

The game $\mathsf{IND\text{-}sPUN\text{-}ATK}_{\mathsf{PE}}^{\mathsf{sel},\mathcal{A}}$ proceeds as follows.

1. **Initialize.** The adversary announces the target tags $\{\widehat{t_1}, \cdots, \widehat{t_d}\}$.
2. **Setup.** The challenger initializes a set punctured tags $\mathcal{T}^* \leftarrow \emptyset$, a counter $i \leftarrow 0$ that counts the current number of punctured tags in $\mathcal{T}^*$ and a set of corrupted tags $\mathcal{C}^* \leftarrow \emptyset$ containing all punctured tags at the time of the first corruption query. Then, it runs $(pk, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda, d)$. Finally, it gives pk to the adversary.
3. **Query 1.**
   - Once the adversary makes a puncture key query $\mathrm{PQ}(t^*)$, the challenger updates $i \leftarrow i+1$, returns $sk_i \leftarrow \mathsf{Punc}(pk, sk_{i-1}, t^*)$ and adds t^* to $\mathcal{T}^*$.
   - The first time the adversary makes a corruption query $\mathrm{CQ}()$, the challenger returns $\perp$ if it finds out that $\{\widehat{t_1}, \cdots, \widehat{t_d}\} \cap \mathcal{T}^* = \emptyset$. Otherwise, the challenger returns the most recent punctured key sk_η, then sets $\mathcal{C}^*$ as the most recent $\mathcal{T}^*$ (i.e., $\mathcal{C}^* \leftarrow \mathcal{T}^* = \{t_1^*, \cdots, t_\eta^*\}$). All subsequent puncture key queries and corruption queries are answered with $\perp$.
   - If $ATK = CCA$: Once the adversary makes a decryption query $\mathrm{DQ}(ct, \{t_1, \cdots, t_d\})$, the challenger runs $\mathsf{Decrypt}(pk, sk_\eta, (ct, \{t_1, \cdots, t_d\}))$ using the most recent punctured key sk_η and returns its output.
     If $ATK = CPA$: the challenger returns $\perp$.
4. **Challenge.** The adversary submits two messages μ_0, μ_1. The challenger rejects the challenge if it finds out that $\{\widehat{t_1}, \cdots, \widehat{t_d}\} \cap \mathcal{C}^* = \emptyset$[3]. Otherwise, the challenger chooses $b \xleftarrow{\$} \{0, 1\}$ and returns $\widehat{ct} \leftarrow \mathsf{Encrypt}(pk, \mu_b, \{\widehat{t_1}, \cdots, \widehat{t_d}\})$.
5. **Query 2.** The same as Query 1 with the restriction that for $\mathrm{DQ}(ct, \{t_1, \cdots, t_d\})$, the challenger returns $\perp$ if $(ct, \{t_1, \cdots, t_d\}) = (\widehat{ct}, \{\widehat{t_1}, \cdots, \widehat{t_d}\})$.
6. **Guess.** The adversary outputs $b' \in \{0, 1\}$. It wins if $b' = b$.

The full security for PE is defined in the same way, except that the adversary can choose target tags at Challenge phase, after getting the public key and after Query 1 phase. In this case, the challenger does not need to check the condition $\{\widehat{t_1}, \cdots, \widehat{t_d}\} \cap \mathcal{T}^* = \emptyset$ in the first corruption query $\mathrm{CQ}()$ of the adversary in Query 1 phase.

[3] Note that, after making some queries that are different from the target tags, the adversary may skip making corruption query but goes directly to the challenge phase and trivially wins the game. This rejection prevents the adversary from such a trivial win. It also force the adversary to make the corruption query before challenging.

2.2 Background on Lattices

In this work, all vectors are written as columns. The transpose of a vector $\mathbf{b}$ (resp., a matrix $\mathbf{A}$) is denoted as $\mathbf{b}^T$ (resp., $\mathbf{A}^T$). The Gram-Schmidt (GS) orthogonaliation of $\mathbf{S} := [\mathbf{s}_1, \cdots, \mathbf{s}_k]$ is denoted by $\widetilde{\mathbf{S}} := [\widetilde{\mathbf{s}}_1, \cdots, \widetilde{\mathbf{s}}_k]$ in the same order.

Lattices. A lattice is a set $\mathcal{L} = \mathcal{L}(\mathbf{B}) := \{\sum_{i=1}^{m} \mathbf{b}_i x_i : x_i \in \mathbb{Z}\ \forall i \in [m]\} \subseteq \mathbb{Z}^m$ generated by a basis $\mathbf{B} = [\mathbf{b}_1 | \cdots | \mathbf{b}_m] \in \mathbb{Z}^{n \times m}$. We are interested in the following lattices:
$\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \text{ s.t. } \mathbf{Ae} = 0 \pmod q\}$, $\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \text{ s.t. } \mathbf{Ae} = \mathbf{u} \pmod{\text{q}}\}$,
$\Lambda_q^{\mathbf{U}}(\mathbf{A}) := \left\{\mathbf{R} \in \mathbb{Z}^{m \times k} \text{ s.t. } \mathbf{AR} = \mathbf{U} (\text{mod q})\right\}$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}^{n \times m}$, $\mathbf{u} \in \mathbb{Z}_q^n$ and $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$.

For a vector $\mathbf{s} = (s_1, \cdots, s_n)$, $\|\mathbf{s}\| := \sqrt{s_1^2 + \cdots + s_n^2}$, $\|\mathbf{s}\|_\infty := \max_{i \in [n]} |s_i|$. For a matrix $\mathbf{S} = [\mathbf{s}_1 \cdots \mathbf{s}_k]$ and any vector $\mathbf{x} = (x_1, \cdots, x_k)$, we define $\|\mathbf{S}\| := \max_{i \in [k]} \|\mathbf{s}_i\|$, the GS norm of $\mathbf{S}$ is $\|\widetilde{\mathbf{S}}\|$, the sup norm is $\|\mathbf{S}\|_{sup} = \sup_{\mathbf{x}} \frac{\|\mathbf{Sx}\|}{\|\mathbf{x}\|}$. This yields for all $\mathbf{x}$ that $\|\mathbf{Sx}\| \leq \|\mathbf{S}\|_{sup} \cdot \|\mathbf{x}\|$. We call a basis $\mathbf{S}$ of some lattice *short* if $\|\widetilde{\mathbf{S}}\|$ is short.

Gaussian Distributions. Assume $m \geq 1$, $\mathbf{v} \in \mathbb{R}^m$, $\sigma > 0$, and $\mathbf{x} \in \mathbb{R}^m$. We define the function $\rho_{\sigma,\mathbf{v}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{v}\|^2 / \sigma^2)$.

Definition 2 (Discrete Gaussians). *Suppose that $\mathcal{L} \subseteq \mathbb{Z}^m$ is a lattice, and $\mathbf{v} \in \mathbb{R}^m$ and $\sigma > 0$. The discrete Gaussian distribution over $\mathcal{L}$ with center $\mathbf{v}$ and parameter σ is defined by $\mathcal{D}_{\mathcal{L},\sigma,\mathbf{v}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{v}}(\mathbf{x})}{\rho_{\sigma,\mathbf{v}}(\mathcal{L})}$ for $\mathbf{x} \in \mathcal{L}$, where $\rho_{\sigma,\mathbf{v}}(\mathcal{L}) := \sum_{\mathbf{x} \in \mathcal{L}} \rho_{\sigma,\mathbf{v}}(\mathbf{x})$.*

Lemma 1 ([23, Lemma 4.4]). *Let $q > 2$ and let $\mathbf{A}, \mathbf{B}$ be a matrix in $\mathbb{Z}_q^{n \times m}$ with $m > n$. Let $\mathbf{T_A}$ be a basis for $\Lambda_q^{\perp}(\mathbf{A})$. Then, for $\sigma \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log n})$, $\Pr[\mathbf{x} \leftarrow \mathcal{D}_{\Lambda_q^{\perp}(\mathbf{A}),\sigma} : \|\mathbf{x}\| > \sigma\sqrt{m}] \leq \mathsf{negl}(n)$.*

Learning with Errors. The security for our construction relies on the decision variant of the learning with errors (DLWE) problem defined below.

Definition 3 (DLWE, [25]). *Suppose that n be a positive integer, q is prime, and χ is a distribution over $\mathbb{Z}_q$. The (n, m, q, χ)-DLWE problem requires to distinguish $(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e})$ from $(\mathbf{A}, \mathbf{c})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{c} \xleftarrow{\$} \mathbb{Z}_q^m$.*

Let χ be a χ_0-bounded noise distribution, i.e., its support belongs to $[-\chi_0, \chi_0]$. The hardness of DLWE is measured by q/χ_0, which is always greater than 1 as χ_0 is chosen such that $\chi_0 < q$. Specifically, the smaller q/χ_0 is, the harder DLWE is. (See [5, Subsection 2.2] and [8, Section 3] for further discussions.)

Lemma 2 ([8, Corollary 3.2]). *For all $\epsilon > 0$, there exist functions $q = q(n) \leq 2^n$, $m = \Theta(n \log q) = \mathsf{poly}(n)$, $\chi = \chi(n)$ such that χ is a χ_0-bounded for some $\chi_0 = \chi_0(n)$, $q/\chi_0 \geq 2^{n^\epsilon}$ and such that $DLWE_{n,m,q,\chi}$ is at least as hard as the classical hardness of $GapSVP_\gamma$ and the quantum hardness of $SIVP_\gamma$ for $\gamma = 2^{\Omega(n^\epsilon)}$.*

The GapSVP_γ problem is the one, given a basis for a lattice and a positive number d, requires to distinguish between two cases; (i)the lattice has a vector shorter than d, and (ii) all lattice vector have length bigger than $\gamma \cdot d$. And SIVP_γ is the problem that, given a basis for a lattice of rannk n, requires to find a set of n "short" and independent lattice vectors.

Leftover Hash Lemma. The following variant of the so-called leftover hash lemma will be used in this work to support our arguments.

Lemma 3 ([1, Lemma 13]). *Let m, n, q be such that $m > (n+1)\log_2 q + \omega(\log n)$ and that $q > 2$ is prime. Let $\mathbf{A}$ and $\mathbf{B}$ are uniformly chosen from $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$, respectively. Then for any uniformly chosen matrix $\mathbf{S}$ from $\{-1,1\}^{m\times k}$ (mod q) and for all vectors $\mathbf{e} \in \mathbb{Z}_q^m$,*

$$(\mathbf{A}, \mathbf{AS}, \mathbf{S}^T\mathbf{e}) \overset{s}{\approx} (\mathbf{A}, \mathbf{B}, \mathbf{S}^T\mathbf{e}).$$

We conclude this section with some standard results regarding trapdoor mechanism often used in lattice-based cryptography.

Lattice Trapdoor Mechanism. In our context, a (lattice) trapdoor is a short basis $\mathbf{T_A}$ for the q-ary lattice $\Lambda_q^\perp(\mathbf{A})$, i.e., $\mathbf{A}\cdot\mathbf{T_A} = 0 \pmod q$ (see [17]). We call $\mathbf{T_A}$ the associated trapdoor for $\Lambda_q^\perp(\mathbf{A})$ or even for $\mathbf{A}$.

Lemma 4. *Let $n, m, q > 0$ and q be prime.*

1. $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$ *([3], [22]): This is a PPT algorithm that outputs a pair $(\mathbf{A}, \mathbf{T_A}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{m\times m}$, where $\mathbf{T_A}$ is a trapdoor for $\Lambda_q^\perp(\mathbf{A})$ such that $\mathbf{A}$ is negligibly close to uniform and $\|\widetilde{\mathbf{T_A}}\| = O(\sqrt{n\log q})$. The algorithm works if $m = \Theta(n\log q)$.*
2. $\mathbf{T_D} \leftarrow \mathsf{ExtBasisRight}(\mathbf{D} := [\mathbf{A}|\mathbf{AS}+\mathbf{B}], \mathbf{T_B})$ *([1]): This is a DPT algorithm that, for the input $(\mathbf{D}, \mathbf{T_B})$, outputs a trapdoor $\mathbf{T_D}$ for $\Lambda_q^\perp(\mathbf{D})$ such that $\|\widetilde{\mathbf{T_D}}\| \le \|\widetilde{\mathbf{T_B}}\|(1 + \|\mathbf{S}\|_{sup})$, where $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$.*
3. $\mathbf{T_E} \leftarrow \mathsf{ExtBasisLeft}(\mathbf{E} := [\mathbf{A}|\mathbf{B}], \mathbf{T_A})$ *([11]): This is a DPT algorithm that for $\mathbf{E}$ of the form $\mathbf{E} := [\mathbf{A}|\mathbf{B}]$ and a trapdoor $\mathbf{T_A}$ for $\Lambda_q^\perp(\mathbf{A})$, outputs a trapdoor $\mathbf{T_E}$ for $\Lambda_q^\perp(\mathbf{E})$ such that $\|\widetilde{\mathbf{T_E}}\| = \|\widetilde{\mathbf{T_A}}\|$, where $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$.*
4. $\mathbf{R} \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{U}, \sigma)$ *([17]): This is a PPT algorithm that takes a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, its associated trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, a matrix $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$ and a real number $\sigma > 0$ and returns a short matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times k}$ chosen randomly according to a distribution that is statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{U}}(\mathbf{A}),\sigma}$. The algorithm works if $\sigma = \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$. Furthermore, $\|\boldsymbol{R}^T\|_{sup} \le \sigma\sqrt{mk}$, $\|\boldsymbol{R}\|_{sup} \le \sigma\sqrt{mk}$ (see also in [5, Lemma 2.5]).*
5. $\mathbf{T'_A} \leftarrow \mathsf{RandBasis}(\mathbf{A}, \mathbf{T_A}, \sigma)$ *([11]): This is a PPT algorithm that takes a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, its associated trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, and a real number $\sigma > 0$ and returns a new basis $\mathbf{T'_A}$ for $\Lambda_q^\perp(\mathbf{A})$ chosen randomly according to a distribution that is statistically close to $(\mathcal{D}_{\Lambda_q^\perp(\mathbf{A}),\sigma})^m$, and $\|\widetilde{\mathbf{T'_A}}\| \le \sigma\sqrt{m}$. The algorithm works if $\sigma = \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$.*

3 Generic PE Construction from DFKHE

3.1 Delegatable Fully Key-homomorphic Encryption

Delegatable fully key-homomorphic encryption (DFKHE) can be viewed as a generalised notion of the so-called fully key-homomorphic encryption (FKHE) [5] augmented with a key delegation mechanism [5].

Informally, FKHE enables one to transform an encryption, say $ct_{\mathbf{x}}$, of a plaintext μ under a public variable $\mathbf{x}$ into the one, say ct_f, of the same μ under some value/function pair (y, f), with the restriction that one is only able to decrypt the ciphertext ct_f if $f(\mathbf{x}) = y$. Similarly, DFHKP together with the key delegation mechanism allows one to do the same but with more functions, i.e., $(y, f_1, \cdots, f_k)$, and the condition for successful decryption is that $f_1(\mathbf{x}) = \cdots = f_k(\mathbf{x}) = y$.

Definition 4 (DFKHE). *Let $\lambda, d = d(\lambda) \in \mathbb{N}$ be two positive integers and let $\mathcal{T} = \mathcal{T}(\lambda)$ and $\mathcal{Y} = \mathcal{Y}(\lambda)$ be two finite sets. Define $\mathcal{F} = \mathcal{F}(\lambda) = \{f | f : \mathcal{T}^d \to \mathcal{Y}\}$ to be a family of efficiently computable functions. $(\lambda, d, \mathcal{T}, \mathcal{Y}, \mathcal{F})$–DFKHE is a tuple consisting of algorithms as follows.*

$(\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk) \leftarrow \mathsf{DFKHE.KGen}(1^\lambda, \mathcal{F})$*: This PPT algorithm takes as input a security parameter λ and outputs a public key* $\mathsf{dfkhe}.pk$ *and a secret key* $\mathsf{dfkhe}.sk$*.*

$\mathsf{dfkhe}.sk_{y,f} \leftarrow \mathsf{DFKHE.KHom}(\mathsf{dfkhe}.sk, (y, f))$*: This PPT algorithm takes as input the secret key* $\mathsf{dfkhe}.sk$ *and a pair $(y, f) \in \mathcal{Y} \times \mathcal{F}$ and returns a secret homomorphic key $sk_{y,f}$.*

$\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_{k+1}} \leftarrow \mathsf{DFKHE.KDel}(\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}, (y, f_{k+1}))$*: This PPT algorithm takes as input the public key* $\mathsf{dfkhe}.pk$*, a function $f_{k+1} \in \mathcal{F}$ and the secret key* $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}$ *and returns the delegated secret key* $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_{k+1}}$*. Further, the key* $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}$ *is produced either by* $\mathsf{DFKHE.KHom}$ *if $k = 1$, or iteratively by* $\mathsf{DFKHE.KDel}$ *if $k > 1$.*

$(\mathsf{dfkhe}.ct, \mathbf{t}) \leftarrow \mathsf{DFKHE.Enc}(\mathsf{dfkhe}.pk, \mu, \mathbf{t})$*: This PPT algorithm takes as input the public key* $\mathsf{dfkhe}.pk$*, a plaintext μ and a variable $\mathbf{t} \in \mathcal{T}^d$ and returns a ciphertext* $\mathsf{dfkhe}.ct$*– an encryption of μ under the variable $\mathbf{t}$.*

$\mathsf{dfkhe}.ct_{f_1,\cdots,f_k} \leftarrow \mathsf{DFKHE.ExtEval}(f_1, \cdots, f_k, (\mathsf{dfkhe}.ct, \mathbf{t}))$*: The DPT algorithm takes as input a ciphertext* $\mathsf{dfkhe}.ct$ *and the associated variable $\mathbf{t} \in \mathcal{T}^d$ and returns an evaluated ciphertext* $\mathsf{dfkhe}.ct_{f_1,\cdots,f_k}$*. If $f_1(\mathbf{t}) = \cdots = f_k(\mathbf{t}) = y$, then we say that* $\mathsf{dfkhe}.ct_{f_1,\cdots,f_k}$ *is an encryption of μ using the public key $(y, f_1, \cdots, f_k)$.*

$\mu/\perp \leftarrow \mathsf{DFKHE.Dec}(\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}, (\mathsf{dfkhe}.ct, \mathbf{t}))$*: The DPT algorithm takes as input a delegated secret key* $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}$ *and a ciphertext* $\mathsf{dfkhe}.ct$ *associated with $\mathbf{t} \in \mathcal{T}^d$ and recovers a plaintext μ. It succeeds if $f_i(\mathbf{t}) = y$ for all $i \in [k]$. Otherwise, it fails and returns $\perp$. To recover μ, the algorithm first calls* $\mathsf{DFKHE.ExtEval}(f_1, \cdots, f_k, (\mathsf{dfkhe}.ct, \mathbf{t}))$ *and gets* $\mathsf{dfkhe}.ct_{f_1,\cdots,f_k}$*. Next it uses* $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}$ *and opens* $\mathsf{dfkhe}.ct_{f_1,\cdots,f_k}$*.*

Obviously, DFKHE from Definition 4 is identical to FKHE [5] if $k = 1$.

Correctness. For all $\mu \in \mathcal{M}$, all $k \in \mathbb{N}$, all $f_1, \cdots, f_k \in \mathcal{F}$ and $\mathbf{t} \in \mathcal{T}^d$, $y \in \mathcal{Y}$, over the randomness of $(\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk) \leftarrow \mathsf{FKHE.KGen}(1^\lambda, \mathcal{F})$, $(\mathsf{dfkhe}.ct, \mathbf{t}) \leftarrow \mathsf{FKHE.Enc}(\mathsf{dfkhe}.pk, \mu, \mathbf{t})$, $\mathsf{dfkhe}.sk_{y,f_1} \leftarrow \mathsf{FKHE.KHom}(\mathsf{dfkhe}.sk, (y, f_1))$ and $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_i} \leftarrow \mathsf{FKHE.KDel}(\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_{i-1}}, (y, f_i))$, $\mathsf{dfkhe}.ct_{f_1,\cdots,f_k} \leftarrow \mathsf{DFKHE.ExtEval}(f_1, \cdots, f_k, (\mathsf{dfkhe}.ct, \mathbf{t}))$ for all $i \in \{2, \cdots, k\}$, then

- $\Pr[\mathsf{FKHE.Dec}(\mathsf{dfkhe}.sk, (\mathsf{dfkhe}.ct, \mathbf{t})) = \mu] \geq 1 - negl(\lambda)$,
- if $y = f_1(\mathbf{t}) = \cdots = f_k(\mathbf{t})$, then

$$\Pr[\mathsf{FKHE.Dec}(\mathsf{dfkhe}.sk, (\mathsf{dfkhe}.ct_{f_1,\cdots,f_k}, \mathbf{t})) = \mu] \geq 1 - negl(\lambda),$$
$$\Pr[\mathsf{FKHE.Dec}(\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_i}, (\mathsf{dfkhe}.ct, \mathbf{t})) = \mu] \geq 1 - negl(\lambda), \forall i \in [k],$$

- For any $i \in [k]$, if $y \neq f_i(\mathbf{t})$,

$$\Pr[\mathsf{FKHE.Dec}(\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_j}, (\mathsf{dfkhe}.ct, \mathbf{t})) = \mu] \leq negl(\lambda), \forall j \in \{i, k\}.$$

Security. Security of DFKHE is similar to that of FKHE from [5] with an extra evaluation that includes the key delegation mechanisms.

Definition 5 (Selectively-secure CPA of DFKHE). *DFKHE is IND-sVAR-CPA if for any polynomial time adversary $\mathcal{B}$ in the game* $\mathsf{IND\text{-}sVAR\text{-}CPA}^{\mathsf{sel},\mathcal{B}}_{\mathsf{DFKHE}}$*, the adversary advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}sVAR\text{-}CPA}}_{\mathsf{DFKHE}}(\mathcal{B}) = |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.

The $\mathsf{IND\text{-}sVAR\text{-}CPA}^{\mathsf{sel},\mathcal{B}}_{\mathsf{DFKHE}}$ game is as follows.

1. **Initialize.** On the security parameter λ and λ–dependent tuple $(d, (\mathcal{T}, \mathcal{Y}, \mathcal{F}))$, $\mathcal{B}$ releases the target variable $\widehat{\mathbf{t}} = (\widehat{t_1}, \cdots, \widehat{t_d}) \in \mathcal{T}^d$.
2. **Setup.** The challenger runs $(\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk) \leftarrow \mathsf{DFKHE.KGen}(1^\lambda, \mathcal{F})$. Then, it gives $\mathsf{dfkhe}.pk$ to $\mathcal{B}$.
3. **Query.** $\mathcal{B}$ adaptively makes delegated key queries $\mathrm{DKQ}(y, (f_1, \cdots, f_k))$ to get the corresponding delegated secret keys. Specifically, $\mathcal{B}$ is allowed to have an access to the oracle $KG(\mathsf{dfkhe}.sk, \widehat{\mathbf{t}}, y, (f_1, \cdots, f_k))$, which takes as input $\mathsf{dfkhe}.sk$, $\widehat{\mathbf{t}}$, a list of functions $f_1, \cdots, f_k \in \mathcal{F}$ and $y \in \mathcal{Y}$ and returns either $\perp$ if all $f_j(\widehat{\mathbf{t}}) = y$, or the delegated secret key $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}$ otherwise. The delegated secret key $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_k}$ is computed calling $\mathsf{dfkhe}.sk_{y,f_1} := \mathsf{DFKHE.KHom}\ (\mathsf{dfkhe}.sk, (y, f_1))$ and $\mathsf{dfkhe}.sk_{y,f_1,\cdots,f_i} \leftarrow \mathsf{DFKHE.KDel}\ (\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk_{y,f_1,\cdots,f_{i-1}}, (y, f_i)),\ \forall i \in \{2, \cdots, k\}$.
4. **Challenge.** The adversary submits two messages μ_0, μ_1 (with $\widehat{\mathbf{t}}$). The challenger in turn chooses $b \xleftarrow{\$} \{0, 1\}$ and returns the output $(\mathsf{dfkhe}.\widehat{ct}, \widehat{\mathbf{t}})$ of $\mathsf{DFKHE.Enc}(\mathsf{dfkhe}.pk, \mu_b, \widehat{\mathbf{t}})$.
5. **Guess.** The adversary outputs $b' \in \{0, 1\}$. It wins if $b' = b$.

3.2 Generic PE Construction from DFKHE.

The main idea behind our construction is an observation that ciphertext tags can be treated as variables $\mathbf{t} = (t_1, \cdots, t_d) \in \mathcal{T}^d$. The puncturing property, which is related to the "equality", suggests us to construct a family $\mathcal{F}$ of equality test functions, allowing to compare each pair of ciphertext tags and punctures. Using this idea, we then can have a PE construction from DFKHE.

Let $\lambda, d = d(\lambda) \in \mathbb{N}$ be two positive integers. Let $\mathcal{T} = \mathcal{T}(\lambda)$ be a finite set (that henceforth called the *tag space*) and $\mathcal{Y} = \mathcal{Y}(\lambda)$ be also a finite set. In addition, let $y_0 \in \mathcal{Y}$ be a some fixed special element. Define a family of all equality test functions indicated by $\mathcal{T}$,

$$\mathcal{F} = \mathcal{F}(\lambda) := \left\{ f_{t^*} | t^* \in \mathcal{T}, \forall \mathbf{t} = (t_1, \cdots, t_d), f_{t^*} : \mathcal{T}^d \to \mathcal{Y} \right\}, \tag{1}$$

where $f_{t^*}(\mathbf{t}) := y_0$ if $t^* \neq t_i, \forall i \in [d]$, and $f_{t^*}(\mathbf{t}) := y_{t^*,\mathbf{t}} \in \mathcal{Y} \setminus \{y_0\}$ otherwise. Here, $y_{t^*,\mathbf{t}}$ means depending on the value of t^* and $\mathbf{t}$. Now, let $\Pi = (\mathsf{DFKHE.KGen}, \mathsf{DFKHE.KHom}, \mathsf{DFKHE.Enc}, \mathsf{DFKHE.ExtEval}, \mathsf{DFKHE.KDel}, \mathsf{DFKHE.Dec})$ be $(\lambda, d, \mathcal{T}, \mathcal{Y}, \mathcal{F})$–DFHKE. Using Π, we can construct a PE system $\Psi = (\mathsf{PE.key}, \mathsf{PE.enc}, \mathsf{PE.pun}, \mathsf{PE.dec})$ of which both tags and punctures reside in $\mathcal{T}$. The description of Ψ is below:

$\underline{(\mathsf{pe}.pk, \mathsf{pe}.sk_0) \leftarrow \mathsf{PE.key}(1^\lambda, d)}$: For input a security parameter λ and the maximum number d of tags per ciphertext, run $(\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk) \leftarrow \mathsf{DFKHE.KGen}(1^\lambda, \mathcal{F})$, and return $\mathsf{pe}.pk := \mathsf{dfkhe}.pk$, and $\mathsf{pe}.sk_0 := \mathsf{dfkhe}.sk$.

$\underline{\mathsf{pe}.ct \leftarrow \mathsf{PE.enc}(\mathsf{pe}.pk, \mu, \mathbf{t} = (t_1, \cdots, t_d))}$: For a public key $\mathsf{pe}.pk$, a message μ, and ciphertext tags $\mathbf{t} = (t_1, \cdots, t_d)$, return $\mathsf{pe}.ct \leftarrow \mathsf{DFKHE.Enc}(\mathsf{pe}.pk, \mu, \mathbf{t})$.

$\underline{\mathsf{pe}.sk_i \leftarrow \mathsf{PE.pun}(\mathsf{pe}.pk, \mathsf{pe}.sk_{i-1}, t_i^*)}$: For input $\mathsf{pe}.pk$, $\mathsf{pe}.sk_{i-1}$ and a punctured tag t_i^*,

- If $i = 1$: run $\mathsf{dfkhe}.sk_{y_0,f_{t_1^*}} \leftarrow \mathsf{DFKHE.KHom}(\mathsf{pe}.sk_0, (y_0, f_{t_1^*}))$ and output $\mathsf{pe}.sk_1 := \mathsf{dfkhe}.sk_{y_0,f_{t_1^*}}$.
- If $i \geq 2$: compute $\mathsf{pe}.sk_i \leftarrow \mathsf{DFKHE.KDel}(\mathsf{dfkhe}.pk, \mathsf{pe}.sk_{i-1}, (y_0, f_{t_i^*}))$.
- Finally, output $\mathsf{pe}.sk_i$.

$\mu/\perp \leftarrow \mathsf{PE.dec}(\mathsf{pe}.pk, (\mathsf{pe}.sk_i, (t_1^*, \cdots, t_i^*)), (\mathsf{pe}.ct, \mathbf{t}))$: For input the public key $\mathsf{pe}.pk$, a puncture key $\mathsf{pe}.sk_i$ together with punctures $(t_1^*, \cdots, t_i^*)$, a ciphertext $\mathsf{pe}.ct$ and its associated tags $\mathbf{t} = (t_1, \cdots, t_d)$, the algorithm first checks whether or not $f_{t_1^*}(\mathbf{t}) = \cdots = f_{t_i^*}(\mathbf{t}) = y_0$. If not, the algorithm returns $\perp$. Otherwise, it returns the output of $\mathsf{DFKHE.Dec}(\mathsf{pe}.sk_i, \mathsf{pe}.ct)$.

Correctness. Remark that, over the choice of $(\lambda, d, \eta, (t_1^*, \cdots, t_\eta^*), (t_1, \cdots, t_d), \eta \geq 0, t_1^*, \cdots, t_\eta^* \in \mathcal{T}$, $t_1, \cdots, t_d \in \mathcal{T} \setminus \{t_1^*, \cdots, t_\eta^*\}$, we have $f_{t_j^*}(\mathbf{t}) = y_0$ for all $j \in [\eta]$. Then, it is clear that, the induced PE Ψ is correct if and only if the DFKHE Π is correct.

Theorem 1. *PE Ψ is selectively-secure CPA assuming that the underlying DFKHE Π is selectively-CPA secure.*

Proof. Assume that there exists an adversary $\mathcal{A}$ that is able to break the selective security of Ψ with probability δ. We can construct a simulator $\mathcal{S}$, which takes advantage of $\mathcal{A}$ and breaks selective security of Π with the same probability.

Initialize. $\mathcal{S}$ would like to break the selective security of the $(\lambda, d, (\mathcal{T}, \mathcal{Y}, \mathcal{F})$–DFKHE system $\Pi = (\mathsf{DFKHE.KGen}, \mathsf{DFKHE.KHom}, \mathsf{DFKHE.Enc}, \mathsf{DFKHE.Dec}, \mathsf{DFKHE.ExtEval}, \mathsf{DFKHE.KDel})$, where $\lambda, d, \mathcal{T}$, $\mathcal{Y}$, $\mathcal{F}$ are specified as in and around Equation (1).

Targeting. $\mathcal{S}$ calls $\mathcal{A}$ to get the target tags $(\widehat{t}_1, \cdots, \widehat{t}_d)$ in the game for Ψ, and lets it be $\widehat{\mathbf{t}}$, playing the role of the target variable in the game for Π.

Setup. $\mathcal{S}$ initializes a set of punctured tags $\mathcal{T}^* \leftarrow \emptyset$, and a set of corrupted tags $\mathcal{C}^* \leftarrow \emptyset$ containing all punctured tags at the time of the first corruption query, runs $(\mathsf{dfkhe}.pp, \mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk) \leftarrow \mathsf{DFKHE.KGen}(1^\lambda, \mathcal{F})$ and gives $\mathsf{dfkhe}.pp, \mathsf{dfkhe}.pk$ to $\mathcal{A}$. Note that $\mathsf{PE.key}(1^\lambda, d) \equiv \mathsf{DFKHE.KGen}(1^\lambda, \mathcal{F})$ by construction.

Query 1. In this phase, $\mathcal{A}$ adaptively makes puncture queries $\mathrm{PQ}(k, t_k^*)$, where k implicitly counts the number of PQ queries so far, and corruption queries CQ(). To reply $\mathrm{PQ}(k, t_k^*)$, $\mathcal{S}$ simply returns the output of $\mathsf{DFKHE.KDel}(\mathsf{dfkhe}.pk, \mathsf{dfkhe}.sk_{y_0, f_{t_1^*}, \cdots, f_{t_{k-1}^*}})$, with noting that when $k = 1$, then we have both $\mathsf{dfkhe}.sk_{y_0, f_{t_1^*}, \cdots, f_{t_{k-1}^*}} := \mathsf{dfkhe}.sk_0$ and $\mathsf{DFKHE.KDel} \equiv \mathsf{DFKHE.KHom}$ and finally appends t_k^* to $\mathcal{T}^*$.
The simulator just cares about the time at which the first CQ() has been completed. At that time, $\mathcal{S}$ saves the value of the counter k and makes $\mathcal{A}$'s puncture queries a list of functions $\{f_{t_1^*}, \cdots, f_{t_k^*}\}$ and sets $\mathcal{C}^* \leftarrow \mathcal{T}^*$. We can consider that $\mathcal{A}$ has made a sequence of k queries to the $KG(\mathsf{dfkhe}.sk, \widehat{\mathbf{t}}, y, (f_1, \cdots, f_k))$ oracle in the DFKHE's security game. Recall that, the requirement for a query to KG to be accepted is that it must be *not* all $j \in [k]$ sastyfying $f_j(\widehat{\mathbf{t}}) = y$. This requirement is essentially fulfilled thanks to the condition in the FE's security game that there is at least one $t_j^* \in \{\widehat{t}_1, \cdots, \widehat{t}_d\} \cap \mathcal{C}^*$.

Challenge. $\mathcal{A}$ submits two messages μ_0, μ_1 (with $\widehat{\mathbf{t}}$). $\mathcal{S}$ in turn chooses $b \xleftarrow{\$} \{0, 1\}$ and returns $(\mathsf{dfkhe}.\widehat{ct}, \widehat{\mathbf{t}}) \leftarrow \mathsf{DFKHE.Enc}(\mathsf{dfkhe}.pk, \mu_b, \widehat{\mathbf{t}})$.

Query 2. The same as Query 1.

Guess. $\mathcal{S}$ outputs the same $b' \in \{0, 1\}$ as $\mathcal{A}$ has guessed.

It is clear that the FE adversary $\mathcal{A}$ is joining the DFKHE game, however it is essentially impossible to distinguish the DFKHE game from the FE one as the simulated environment for $\mathcal{A}$ is *perfect*. This concludes the proof. □

4 DFKHE and FE Construction from Lattices

At first, in Subsection 4.1 below, we will review the key-homomorphic mechanism, which is an important ingredient for our lattice-based construction.

4.1 Key-homomorphic Mechanism for Arithmetic Circuits

Let $n, q > 0$, $k := \lceil \log q \rceil$ and $m := n \cdot k$. We exploit the gadget matrix $\mathbf{G}$ and its associated trapdoor $\mathbf{T_G}$. According to [22, Section 4], the matrix $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^T \in \mathbb{Z}_q^{n \times m}$, where $\mathbf{g}^T = [1\ 2\ 4\ \cdots\ 2^{k-1}]$. The associated trapdoor $\mathbf{T_G} \in \mathbb{Z}^{m \times m}$ is publicly known and $\|\widetilde{\mathbf{T_G}}\| \leq \sqrt{5}$ (see [22, Theorem 4.1]).

Key-homomorphic Mechanism. We recap some basic facts useful for construction of evaluation algorithms for the family of polynomial depth and unbounded fan-in arithmetic circuits (see [5, Section 4] for details). Let $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ be the gadget matrix given above. For $x \in \mathbb{Z}_q$, $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$, define the following set $E_{\mathbf{s},\delta}(x, \mathbf{B}) := \{(x\mathbf{G} + \mathbf{B})^T\mathbf{s} + \mathbf{e},\ \text{where } \|\mathbf{e}\| < \delta\}$. More details can be found in [5].

Lemma 5 ([5, Section 4]). *Let n, $q = q(n)$, $m = \Theta(n \log q)$ be positive integers, $\mathbf{x} = (x_1, \cdots, x_d) \in \mathbb{Z}_q^d$, $\mathbf{x}^* = (x_1^*, \cdots, x_d^*) \in \mathbb{Z}_q^d$, $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, $\mathbf{c}_i \in E_{\mathbf{s},\delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$, $\mathbf{S}_i \in \mathbb{Z}_q^{m \times m}$ for all $i \in [d]$. Also, let $\beta_{\mathcal{F}} = \beta_{\mathcal{F}}(n) : \mathbb{Z} \to \mathbb{Z}$ be a positive integer-valued function, and $\mathcal{F} = \{f : (\mathbb{Z}_q)^d \to \mathbb{Z}_q\}$ be a family of functions, in which each function can be computed by some circuit of a family of depth τ, polynomial-size arithmetic circuits $(C_\lambda)_{\lambda \in \mathbb{N}}$. Then there exist DPT algorithms $\mathsf{Eval_{pk}}$, $\mathsf{Eval_{ct}}$, $\mathsf{Eval_{sim}}$ associated with $\beta_{\mathcal{F}}$ and $\mathcal{F}$ such that the following properties hold.*

1. *If $\mathbf{B}_f \leftarrow \mathsf{Eval_{pk}}(f \in \mathcal{F}, (\mathbf{B}_i)_{i=1}^d)$, then $\mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$.*
2. *Let $\mathbf{c}_f \leftarrow \mathsf{Eval_{ct}}(f \in \mathcal{F}, ((x_i, \mathbf{B}_i, \mathbf{c}_i))_{i=1}^d)$, then $\mathbf{c}_f \in E_{\mathbf{s},\Delta}(f(\mathbf{x}), \mathbf{B}_f)$, in which $\mathbf{B}_f \leftarrow \mathsf{Eval_{pk}}(f, (\mathbf{B}_i)_{i=1}^d)$ and $\Delta < \delta \cdot \beta_{\mathcal{F}}$.*
3. *The output $\mathbf{S}_f \leftarrow \mathsf{Eval_{sim}}(f \in \mathcal{F}, ((x_i^*, \mathbf{S}_i))_{i=1}^d, \mathbf{A})$ satisfies the relation $\mathbf{A}\mathbf{S}_f - f(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_f$ and $\|\mathbf{S}_f\|_{sup} < \beta_{\mathcal{F}}$ with overwhelming probability, where $\mathbf{B}_f \leftarrow \mathsf{Eval_{pk}}(f, (\mathbf{A}\mathbf{S}_i - x_i^*\mathbf{G})_{i=1}^d)$. In particular, if $\mathbf{S}_1, \cdots, \mathbf{S}_d \xleftarrow{\$} \{-1, 1\}^{m \times m}$, then $\|\mathbf{S}_f\|_{sup} < \beta_{\mathcal{F}}$ with all but negligible probability for all $f \in \mathcal{F}$.*

In general, for a family $\mathcal{F}$ of functions represented by polynomial-size and unbounded fan-in circuits of depth τ, the function $\beta_{\mathcal{F}}$ is given by the following lemma.

Lemma 6 ([5, Lemma 5.3]). *Let n, $q = q(n)$, $m = \Theta(n \log q)$ be positive integers. Let $\mathcal{C}_\lambda$ be a family of polynomial-size arithmetic circuits of depth τ and $\mathcal{F} = \{f : (\mathbb{Z}_q)^d \to \mathbb{Z}_q\}$ be the set of functions f that can be computed by some circuit $\mathcal{C} \in \mathcal{C}_\lambda$ as stated in Lemma 5. Also, suppose that all (but possibly one) of the input values to the multiplication gates are bounded by $p < q$. Then, $\beta_{\mathcal{F}} = (\frac{p^d - 1}{p - 1} \cdot m)^\tau \cdot 20\sqrt{m} = O((p^{d-1}m)^\tau \sqrt{m})$.*

Definition 6 (FKHE enabling functions). *The tuple $(\mathsf{Eval_{pk}}, \mathsf{Eval_{ct}}, \mathsf{Eval_{sim}})$ together with the family $\mathcal{F}$ and the function $\beta_{\mathcal{F}} = \beta_{\mathcal{F}}(n)$ in the Lemma 5 is called $\beta_{\mathcal{F}}$-FKHE enabling for the family $\mathcal{F}$.*

4.2 LWE-based DFKHE Construction

Our LWE-based DFKHE construction Π is adapted from LWE–based FKHE and the key delegation mechanism, both of which proposed in [5]. Roughly speaking, the key delegation mechanism in the lattice setting is triggered using the algorithms $\mathsf{ExtBasisLeft}$ and $\mathsf{ExtBasisRight}$ and $\mathsf{RandBasis}$ in Lemma 4. Formally, LWE-based DFKHE Π consists of the following algorithms:

Parameters: Let $\lambda \in \mathbb{N}$ be a security parameter. Set $n = n(\lambda)$, $q = q(\lambda)$ and $d = d(\lambda)$ to be fixed such that $d < q$. Let $\eta \in \mathbb{N}$ be the maximum number of variables that can be delegated and $\sigma_1, \cdots, \sigma_\eta$ be Gaussian parameters. Also, we choose a constant $\epsilon \in (0, 1)$, which is mentioned in Lemma 2. The constant is used to determine the tradeoff between the security level and the efficiency of the system. Let $\mathcal{F} := \{f | f : (\mathbb{Z}_q)^d \to \mathbb{Z}_q\}$ be a family of efficiently computable functions over $\mathbb{Z}_q$ that can be computed by some circuit of a family of depth τ, polynomial-size arithmetic circuits $(C_\lambda)_{\lambda \in \mathbb{N}}$. Take the algorithms ($\mathsf{Eval_{pk}}$, $\mathsf{Eval_{ct}}$, $\mathsf{Eval_{sim}}$) together with a function $\beta_\mathcal{F} = \beta_\mathcal{F}(n)$ to be $\beta_\mathcal{F}$–FKHE enabling for $\mathcal{F}$.

$\mathsf{DFKHE.KGen}(1^\lambda, \mathcal{F})$: For the input pair (a security parameter $\lambda \in \mathbb{N}$ and a family $\mathcal{F}$) [4], do the following:

1. Choose $m = \Theta(n \log q)$. The plaintext space is $\mathcal{M} := \{0, 1\}^m$, $\mathcal{T} := \mathbb{Z}_q$. Additionally, let χ be a χ_0–bounded noise distribution (i.e, its support belongs to $[-\chi_0, \chi_0]$) for which the $(n, 2m, q, \chi)$–DLWE is hard.
2. Generate $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$, sample $\mathbf{U}, \mathbf{B}_1, \cdots, \mathbf{B}_d \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.
3. Output the public key $pk = \{\mathbf{A}, \mathbf{B}_1, \cdots \mathbf{B}_d, \mathbf{U}\}$ and the initial secret key $sk = \{\mathbf{T_A}\}$.

$\mathsf{DFKHE.KHom}(sk, (y, f_1))$: For the input pair (the initial secret key sk and a pair $(y, f_1) \in \mathbb{Z}_q \times \mathcal{F}$) do the following:

1. $\mathbf{B}_{f_1} \leftarrow \mathsf{Eval_{pk}}(f_1, (\mathbf{B}_k)_{k=1}^d)$, $\mathbf{E}_{y,f_1} \leftarrow \mathsf{ExtBasisLeft}([\mathbf{A}|y\mathbf{G} + \mathbf{B}_{f_1}], \mathbf{T_A})$.
2. $\mathbf{T}_{y,f_1} \leftarrow \mathsf{RandBasis}([\mathbf{A}|y\mathbf{G} + \mathbf{B}_{f_1}], \mathbf{E}_{y,f_1}, \sigma_1)$, output the secret key $sk_{y,f_1} = \{\mathbf{T}_{y,f_1}\}$. Here, we set $\sigma_1 = \omega(\beta_\mathcal{F} \cdot \sqrt{\log(2m)})$ for the security proof to work.

$\mathsf{DFKHE.KDel}(sk_{y,f_1,\cdots,f_{\eta-1}}, (y, f_\eta))$: For the input pair (the delegated secret key $sk_{y,f_1,\cdots,f_{\eta-1}}$ and a pair $(y, f_\eta) \in \mathbb{Z}_q \times \mathcal{F}$) do the following:

1. $\mathbf{B}_{f_\eta} \leftarrow \mathsf{Eval_{pk}}(f_\eta, (\mathbf{B}_k)_{k=1}^d)$.
2. $\mathbf{E}_{y,f_1,\cdots,f_\eta} \leftarrow \mathsf{ExtBasisLeft}([\mathbf{A}|y\mathbf{G} + \mathbf{B}_{f_1}| \cdots |y\mathbf{G} + \mathbf{B}_{f_{\eta-1}}|y\mathbf{G} + \mathbf{B}_{f_\eta}], \mathbf{T}_{y,f_1,\cdots,f_{\eta-1}})$.
3. $\mathbf{T}_{y,f_1,\cdots,f_\eta} \leftarrow \mathsf{RandBasis}([\mathbf{A}|y\mathbf{G} + \mathbf{B}_{f_1}| \cdots |y\mathbf{G} + \mathbf{B}_{f_{\eta-1}}|y\mathbf{G} + \mathbf{B}_{f_\eta}], \mathbf{E}_{y,f_1,\cdots,f_\eta}, \sigma_\eta)$.
4. Output the secret key $sk_{y,f_1,\cdots,f_\eta} = \{\mathbf{T}_{y,f_1,\cdots,f_\eta}\}$.

We set $\sigma_\eta = \sigma_1 \cdot (\sqrt{m \log m})^{\eta-1}$ and discuss on setting parameters in details later.

$\mathsf{DFKHE.Enc}(\mu, pk, \mathbf{t})$: For the input consiting of (a message $\mu = (\mu_1, \cdots, \mu_m) \in \mathcal{M}$, the public key pk and ciphertext tags $\mathbf{t} = (t_1, \cdots, t_d) \in \mathcal{T}^d$), perform the following steps:

1. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_{\mathsf{out}}, \mathbf{e}_{\mathsf{in}} \leftarrow \chi^m$, and $\mathbf{S}_1, \cdots, \mathbf{S}_d \xleftarrow{\$} \{-1, 1\}^{m \times m}$.
2. Compute $\mathbf{e} \leftarrow (\mathbf{I}_m|\mathbf{S}_1| \cdots |\mathbf{S}_d)^T \mathbf{e}_{\mathsf{in}} = (\mathbf{e}_{\mathsf{in}}^T, \mathbf{e}_1^T, \cdots, \mathbf{e}_d^T)^T$.
3. Form $\mathbf{H} \leftarrow [\mathbf{A}|t_1\mathbf{G} + \mathbf{B}_1| \cdots |t_d\mathbf{G} + \mathbf{B}_d]$ and compute $\mathbf{c} = \mathbf{H}^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^{(d+1)m}$, $\mathbf{c} = [\mathbf{c}_{\mathsf{in}}|\mathbf{c}_1| \cdots |\mathbf{c}_d]$, where $\mathbf{c}_{\mathsf{in}} = \mathbf{A}^T\mathbf{s} + \mathbf{e}_{\mathsf{in}}$ and $\mathbf{c}_i = (t_i\mathbf{G} + \mathbf{B}_i)^T\mathbf{s} + \mathbf{e}_i$ for $i \in [d]$.
4. Compute $\mathbf{c}_{\mathsf{out}} \leftarrow \mathbf{U}^T\mathbf{s} + \mathbf{e}_{\mathsf{out}} + \mu\lceil\frac{q}{2}\rceil$.
5. Output the ciphertext $(ct_\mathbf{t} = (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_1, \cdots, \mathbf{c}_d, \mathbf{c}_{\mathsf{out}}), \mathbf{t})$.

$\mathsf{DFKHE.ExtEval}(f_1, \cdots, f_\eta, ct_\mathbf{t})$: For the input (a ciphertext $ct_\mathbf{t} = (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_1, \cdots, \mathbf{c}_d, \mathbf{c}_{\mathsf{out}})$ and its associated tags $\mathbf{t} = (t_1, \cdots, t_d)$, and a list of functions $f_1, \cdots, f_\eta \in \mathcal{F}$), execute the following steps:

1. Evaluate $\mathbf{c}_{f_j} \leftarrow \mathsf{Eval_{ct}}(f_j, ((t_k, \mathbf{B}_k, \mathbf{c}_k))_{k=1}^d)$ for $j \in [\eta]$.
2. Output the evaluated ciphertext $\mathbf{c}_{f_1,\cdots,f_\eta} := (\mathbf{c}_{f_1}, \cdots, \mathbf{c}_{f_\eta})$.

$\mathsf{DFKHE.Dec}(ct_\mathbf{t}, sk_{y,f_1,\cdots,f_\eta})$: For the input (a ciphertext $ct_\mathbf{t} = (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_1, \cdots, \mathbf{c}_d, \mathbf{c}_{\mathsf{out}})$, the associated tags $\mathbf{t} = (t_1, \cdots, t_d)$, and a delegated secret key $sk_{y,f_1,\cdots,f_\eta}$, execute the following steps:

1. If $\exists j \in [\eta]$ s.t. $f_j(\mathbf{t}) \neq y$, then output $\perp$. Otherwise, go to Step 2.
2. Sample $\mathbf{R} \leftarrow \mathsf{SampleD}([\mathbf{A}|y\mathbf{G} + \mathbf{B}_{f_1}| \cdots |y\mathbf{G} + \mathbf{B}_{f_\eta}], \mathbf{T}_{y,f_1,\cdots,f_\eta}, \mathbf{U}, \sigma_\eta)$.

[4] Here, d also appears implicitly as an input.

3. Evaluate $(\mathbf{c}_{f_1}, \cdots, \mathbf{c}_{f_\eta}) \leftarrow \mathsf{DFKHE.ExtEval}(f_1, \cdots, f_\eta, ct_{\mathbf{t}})$.
4. Compute $\bar{\mu} := (\bar{\mu}_1, \cdots, \bar{\mu}_m) \leftarrow \mathbf{c}_{\mathsf{out}} - \mathbf{R}^T(\mathbf{c}_{\mathsf{in}}|\mathbf{c}_{f_1}|\cdots|\mathbf{c}_{f_\eta})$.
5. For $\ell \in [m]$, if $|\bar{\mu}_\ell| < q/4$ then output $\mu_\ell = 0$; otherwise, output $\mu_\ell = 1$.

In the following, we will demonstrate the correctness and the security of the LWE-based DFKHE Π.

Theorem 2 (Correctness of Π). *The proposed DFKHE Π is correct if the condition*

$$(\eta+1)^2 \cdot \sqrt{m} \cdot \omega((\sqrt{m \log m})^\eta) \cdot \beta_{\mathcal{F}}^2 + 2 < \frac{1}{4}(q/\chi_0) \tag{2}$$

holds, assumming that $f_j(\mathbf{t}) = y$ for all $j \in [\eta]$.

Proof. We have $\bar{\mu} = \mathbf{c}_{\mathsf{out}} - \mathbf{R}^T(\mathbf{c}_{\mathsf{in}}|\mathbf{c}_{f_1}|\cdots|\mathbf{c}_{f_\eta}) = \mu\lceil\frac{q}{2}\rceil + \mathbf{e}_{\mathsf{out}} - \mathbf{R}^T(\mathbf{e}_{\mathsf{in}}|\mathbf{e}_{f_1}|\cdots|\mathbf{e}_{f_\eta})$. Next, we evaluate the norm of $e_{\mathsf{out}} - \mathbf{R}^T(\mathbf{e}_{\mathsf{in}}|\mathbf{e}_{f_1}|\cdots|\mathbf{e}_{f_\eta})$. Since $\mathbf{c}_{f_j} \in E_{\mathbf{s},\Delta}(y, \mathbf{B}_{f_\eta})$, for all $j \in [\eta]$, where $\Delta < \chi_0 \cdot \beta_{\mathcal{F}}$, then $\|(\mathbf{e}_{\mathsf{in}}|\mathbf{e}_{f_1}|\cdots|\mathbf{e}_{f_\eta})\| \le \eta \cdot \Delta + \chi_0 \le (\eta \cdot \beta_{\mathcal{F}} + 1)\chi_0$. Then

$$\begin{aligned}\|\mathbf{e}_{\mathsf{out}} - \mathbf{R}^T(\mathbf{e}_{\mathsf{in}}|\mathbf{e}_{f_1}|\cdots|\mathbf{e}_{f_\eta})\|_\infty &\le \|\mathbf{e}_{\mathsf{out}}\|_\infty + \|\mathbf{R}^T\|_{sup} \cdot \|(\mathbf{e}_{\mathsf{in}}|\mathbf{e}_{f_1}|\cdots|\mathbf{e}_{f_\eta})\| \\ &\le ((\eta+1)^2 \cdot \sqrt{m} \cdot \omega((\sqrt{m \log m})^\eta) \cdot \beta_{\mathcal{F}}^2 + 2) \cdot \chi_0,\end{aligned}$$

where $\|\mathbf{R}^T\|_{sup} \le (\eta+1)m\sigma_\eta$ by Item 4 of Lemma 4 and $\sigma_\eta = \sigma_1 \cdot (\sqrt{m \log m})^{\eta-1} = \omega(\beta_{\mathcal{F}} \cdot \sqrt{\log m}) \cdot (\sqrt{m \log m})^{\eta-1}$.

By choosing parameters such that $((\eta+1)^2 \cdot \sqrt{m} \cdot \omega((\sqrt{m \log m})^\eta) \cdot \beta_{\mathcal{F}}^2 + 2) \cdot \chi_0 < q/4$, which yields Equation (2), then the decryption is successful. □

Theorem 3 (IND-sVAR-CPA of Π). *Assuming the hardness of $(n, 2m, q, \chi)$–DLWE, the proposed DFKHE Π is IND-sVAR-CPA.*

Proof. The proof consists of a sequence of four games, in which the first Game 0 is the original $\mathsf{IND\text{-}sVAR\text{-}CPA}_{\Psi}^{\mathsf{sel},\mathcal{A}}$ game. The last game chooses the challenge ciphertext uniformly at random. Hence, the advantage of the adversary in the last game is zero. The games 2 and 3 are indistinguishable thanks to a reduction from the DLWE hardness.

Game 0. This is the original $\mathsf{IND\text{-}sVAR\text{-}CPA}_{\Psi}^{\mathsf{sel},\mathcal{A}}$ game being played by an adversary $\mathcal{A}$ and a challenger. At the initial phase, $\mathcal{A}$ announces a target variable $\widehat{\mathbf{t}} = (\widehat{t_1}, \cdots, \widehat{t_d})$. Note that, the challenger has to reply delegate key queries $\mathrm{DKQ}(y, f_1, \cdots, f_k)$. However, if $(y, (f_1, \cdots, f_k)) \in \mathbb{Z}_q \times \mathcal{F}^k$ such that $f_1(\widehat{\mathbf{t}}) = \cdots = f_1(\widehat{\mathbf{t}}) = y$ then the query will be aborted. At the setup phase, the challenger generates $pk = \{\mathbf{A}, \mathbf{B}_1, \cdots \mathbf{B}_d, \mathbf{U}\}$, the initial secret key $sk = \{\mathbf{T}_{\mathbf{A}}\}$, where $\mathbf{B}_1, \cdots \mathbf{B}_d \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(n, m, q)$. The challenger then sends pk to the adversary, while it keeps sk secret. Also, in order to produce the challenge ciphertext $\widehat{ct}$ in the challenge phase, $\widehat{\mathbf{S}}_1, \cdots, \widehat{\mathbf{S}}_d \in \{-1, 1\}^{m \times m}$ are generated (Step 2 of $\mathsf{DFKHE.Enc}$).

Game 1. This game slightly changes the way $\mathbf{B}_0, \cdots \mathbf{B}_d$ are generated in the setup phase. Instead in the challenge phase, $\widehat{\mathbf{S}}_1, \cdots, \widehat{\mathbf{S}}_d \in \{-1, 1\}^{m \times m}$ are sampled in the setup phase. This allows to compute $\mathbf{B}_i := \mathbf{A}\widehat{\mathbf{S}}_i - \widehat{t_i}\mathbf{G}$ for $i \in [d]$. The rest of the game is the same as Game 0.

Game 1 and Game 0 are indistinguishable thanks to the leftover hash lemma (i.e., Lemma 3).

Game 2. In this game, the matrix $\mathbf{A}$ is not generated by $\mathsf{TrapGen}$ but chosen uniformly at random from $\mathbb{Z}_q^{n\times m}$. The matrices $\mathbf{B}_1, \cdots \mathbf{B}_d$ are constructed as in Game 1. The secret key is $sk_0 = \{\mathbf{T_G}\}$ instead.

The challenger replies to a delegated key query $\mathrm{DKQ}(y, f_1, \cdots, f_k)$ as follows:

1. If $f_1(\widehat{\mathbf{t}}) = f_k(\widehat{\mathbf{t}}) = y$, the challenger aborts and restarts the game until there exists at least one $f_j(\widehat{\mathbf{t}}) \neq y$. Without loss of generality, we can assume that $f_k(\widehat{\mathbf{t}}) \neq y$.
2. For all $i \in [k]$, compute $\widehat{\mathbf{S}}_{f_i} \leftarrow \mathsf{Eval_{sim}}(f_i, ((\widehat{t_j}, \widehat{\mathbf{S}_j}))_{j=1}^d, \mathbf{A})$, and let $\mathbf{B}_{fi} = \mathbf{A}\widehat{\mathbf{S}}_{f_i} - f_i(\widehat{\mathbf{t}})\mathbf{G}$. Remark that, $\mathbf{B}_{f_1} = \mathsf{Eval_{pk}}(f_i, (\mathbf{B}_j)_{j=1}^d)$. For choosing Gaussian parameters, note that $\|\widehat{\mathbf{S}}_{f_i}\|_{sup} \leq \beta_{\mathcal{F}}$ due to Item 3 of Lemma 5.
3. $\mathbf{E}_{y,f_1,\cdots,f_k} \leftarrow \mathsf{ExtBasisRight}([\mathbf{A}|\mathbf{A}\widehat{\mathbf{S}}_{f_1} + (y - f_1(\widehat{\mathbf{t}}))\mathbf{G}| \cdots |\mathbf{A}\widehat{\mathbf{S}}_{f_k} + (y - f_k(\widehat{\mathbf{t}}))\mathbf{G}], \mathbf{T_G})$. Note that, $\|\mathbf{E}_{y,f_1,\cdots,f_k}\| \leq \|\widetilde{\mathbf{T_G}}\|(1 + \|\mathbf{S}_{f_k}\|_{sup}) = \sqrt{5}(1+\beta_{\mathcal{F}})$ for all $k \in [\eta]$ by Item 2 of Lemma 4.
4. $\mathbf{T}_{y,f_1,\cdots,f_k} \leftarrow \mathsf{RandBasis}([\mathbf{A}|\mathbf{A}\widehat{\mathbf{S}}_{f_1} + (y - f_1(\widehat{\mathbf{t}}))\mathbf{G}| \cdots |\mathbf{A}\widehat{\mathbf{S}}_{f_k} + (y - f_k(\widehat{\mathbf{t}}))\mathbf{G}], \mathbf{E}_{y,f_1,\cdots,f_k}, \sigma_k)$.
5. Return $sk_{y,f_1,\cdots,f_k} := \{\mathbf{T}_{y,f_1,\cdots,f_k}\}$.

Game 2 and Game 1 are indistinguishable. The reason is that the distributions of $\mathbf{A}$'s in both games are statistically close and that the challenger's response to the adversary's query is also the output of $\mathsf{RandBasis}$.

Game 3. This game is similar to Game 2, except that the challenge ciphertext $\widehat{ct}$ is chosen randomly. Therefore, the advantage of the adversary $\mathcal{A}$ in Game 3 is zero.

Now we show that Games 2 and 3 are indistinguishable using a reduction from DLWE.

Reduction from DLWE. Suppose that $\mathcal{A}$ can distinguish Game 2 from Game 3 with a non-negligible advantage. Using $\mathcal{A}$, we construct a DLWE solver $\mathcal{B}$. The reduction is as follows:

- $(n, 2m, q, \chi)$–**DLWE instance.** $\mathcal{B}$ is given a $\mathbf{F} \xleftarrow{\$} \mathbb{Z}_q^{n\times 2m}$, and a vector $\mathbf{c} \in \mathbb{Z}_q^{2m}$, where either (i) $\mathbf{c}$ is random or (ii) $\mathbf{c}$ is in the LWE form $\mathbf{c} = \mathbf{F}^T\mathbf{s} + \mathbf{e}$,
  for some random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \chi^{2m}$. The goal of $\mathcal{B}$ is to decide whether $\mathbf{c}$ is random or generated from LWE.
- **Initial.** $\mathcal{B}$ now parses $[\mathbf{c}_{\mathsf{in}}^T|\mathbf{c}_{\mathsf{out}}^T]^T \leftarrow \mathbf{c}$, where $\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{out}} \in \mathbb{Z}_q^m$, $[\mathbf{e}_{\mathsf{in}}^T|\mathbf{e}_{\mathsf{out}}^T]^T \leftarrow \mathbf{e}$, where $\mathbf{e}_{\mathsf{in}}, \mathbf{e}_{\mathsf{out}} \leftarrow \chi^m$, and $[\mathbf{A}|\mathbf{U}] \leftarrow \mathbf{F}$, where $\mathbf{A}, \mathbf{U} \in \mathbb{Z}_q^{n\times m}$. That is,

$$\mathbf{c}_{\mathsf{in}} = \mathbf{A}^T\mathbf{s} + \mathbf{e}_{\mathsf{in}}, \quad \mathbf{c}_{\mathsf{out}} = \mathbf{U}^T\mathbf{s} + \mathbf{e}_{\mathsf{out}}. \tag{3}$$

  Now $\mathcal{B}$ calls $\mathcal{A}$ to get the target variable $\widehat{\mathbf{t}} = (\widehat{t_1}, \cdots, \widehat{t_d})$ to be challenged.
- **Setup.** $\mathcal{B}$ generates the keys as in Game 2. That is, $\widehat{\mathbf{S}}_1, \cdots, \widehat{\mathbf{S}}_d \xleftarrow{\$} \{-1, 1\}^{m\times m}$ and $\mathbf{B}_i := \mathbf{A}\widehat{\mathbf{S}}_i - \widehat{t_i}\mathbf{G}$ for $i \in [d]$. Finally, $\mathcal{B}$ sends $\mathcal{A}$ the public key $pk = (\mathbf{A}, \mathbf{B}_1, \cdots, \mathbf{B}_d, \mathbf{U})$. Also, $\mathcal{B}$ keeps $sk = \{\mathbf{T_G}\}$ as the initial secret key.
- **Query.** Once $\mathcal{A}$ makes a delegated key query, $\mathcal{B}$ replies as in Game 2.
- **Challenge.** Once $\mathcal{A}$ submits two messages μ_0 and μ_1, $\mathcal{B}$ chooses uniformly at random $b \xleftarrow{\$} \{0, 1\}$, then computes $\widehat{\mathbf{c}} \leftarrow [\mathbf{I}_m|\widehat{\mathbf{S}}_1| \cdots |\widehat{\mathbf{S}}_d]^T\mathbf{c}_{\mathsf{in}} \in \mathbb{Z}_q^{(d+1)m}$ and $\widehat{\mathbf{c}}_{\mathsf{out}} \leftarrow \mathbf{c}_{\mathsf{out}} + \mu_b\lceil\frac{q}{2}\rceil \in \mathbb{Z}_q$.
  - Suppose $\mathbf{c}$ is generated by LWE, i.e., $\mathbf{c}_{\mathsf{in}}$, $\mathbf{c}_{\mathsf{out}}$ satisfy Equation (3). In the $\mathsf{DFKHE.Enc}$ algorithm, $\mathbf{H} = [\mathbf{A}|\widehat{t_1}\mathbf{G} + \mathbf{B}_1| \cdots |\widehat{t_d}\mathbf{G} + \mathbf{B}_d] = [\mathbf{A}|\mathbf{A}\widehat{\mathbf{S}}_1| \cdots |\mathbf{A}\widehat{\mathbf{S}}_d]$. Then

$$\widehat{\mathbf{c}} = [\mathbf{I}_m|\widehat{\mathbf{S}}_1| \cdots |\widehat{\mathbf{S}}_d]^T(\mathbf{A}^T\mathbf{s} + \mathbf{e}_{\mathsf{in}}) = \mathbf{H}^T\mathbf{s} + \widehat{\mathbf{e}},$$

    where $\widehat{\mathbf{e}} = [\mathbf{I}_m|\widehat{\mathbf{S}}_1| \cdots |\widehat{\mathbf{S}}_d]^T\mathbf{e}_{\mathsf{in}}$. It is easy to see that $\widehat{\mathbf{c}}$ is computed as in Game 2. Additionally, $\widehat{\mathbf{c}}_{\mathsf{out}} = \mathbf{U}^T\mathbf{s} + \widehat{\mathbf{e}}_{\mathsf{out}} + \mu_b\lceil\frac{q}{2}\rceil \in \mathbb{Z}_q$. Then $\widehat{ct} := (\widehat{\mathbf{c}}, \widehat{\mathbf{c}}_{\mathsf{out}}) \in \mathbb{Z}_q^{(d+2)m}$ is a valid ciphertext of μ_b.

- If $\mathbf{c}_{\mathsf{in}}$, $\mathbf{c}_{\mathsf{out}}$ are random then $\widehat{\mathbf{c}}$ is random (following a standard left over hash lemma argument). And since $\widehat{\mathbf{c}}_{\mathsf{out}}$ is also random, $\widehat{ct} := (\widehat{\mathbf{c}}, \widehat{\mathbf{c}}_{\mathsf{out}})$ is random in $\mathbb{Z}_q^{(d+2)m}$ which behaves similarly to Game 3.

– **Guess.** Eventually, once $\mathcal{A}$ outputs his guess of whether he is interacting with Game 2 or Game 3, $\mathcal{B}$ outputs his decision for the DLWE problem.

We have shown that $\mathcal{B}$ can solve the $(n, 2m, q, \chi)$–DLWE instance. □

Setting Parameters. In order to choose parameters, we should take the following into consideration:

– For the hardness of DLWE, by Theorem 2, we choose ϵ, n, q, χ, where χ is a χ_0-bounded distribution, such that $q/\chi_0 \geq 2^{n^\epsilon}$. We also note that, the hardness of DLWE via the traditional worst-case reduction (e.g., Lemma 2) does not help us much in proposing concrete parameters for lattice-based cryptosystems. Instead, a more conservative methodology that has been usually used in the literature is the so-called "core-SVP hardness"; see [2, Subsection 5.2.1] for a detailed reference.

– Setting Gaussian parameters:
  1. *First approach:* Without caring the security proof, for trapdoor algorithms to work, we can set $\sigma_1 = \|\widetilde{\mathbf{T}_{\mathbf{A}}}\| \cdot \omega(\sqrt{\log(2m)})$, with $\|\widetilde{\mathbf{T}_{\mathbf{A}}}\| = O(\sqrt{n \log m})$ by Item 1 of Lemma 4. Note that, in DFKHE.KHom we have $\|\widetilde{\mathbf{T}}_{y,f_1}\| < \sigma_1 \cdot \sqrt{2m}$ by Item 5 of Lemma 4. Then, $\sigma_2 = \|\widetilde{\mathbf{T}}_{y,f_1}\| \cdot \omega(\sqrt{\log(3m)}) = \sigma_1 \cdot \omega(\sqrt{m \log m})$. Similarly, we can set $\sigma_k = \sigma_1 \cdot (\sqrt{m \log m})^{k-1}$ for all $k \in [\eta]$.
  2. *Second approach:* For the security proof to work, we have to be careful in choosing Gaussian parameters $\sigma_1, \cdots, \sigma_\eta$. Indeed, we have to choose $\sigma_1 = \omega(\beta_{\mathcal{F}} \cdot \sqrt{\log m})$. In fact, we remarked in Step 2 of **Game 2** of the proof for Theorem 3 that $\|\widehat{\mathbf{S}}_{f_i}\|_{sup} \leq \beta_{\mathcal{F}}$ for all i. And for a generic k we still obtain $\|\widetilde{\mathbf{E}}_{y,f_1,\cdots,f_k}\| \leq \|\widetilde{\mathbf{T}_{\mathbf{G}}}\|(1 + \|\mathbf{S}_{f_k}\|_{sup}) = \sqrt{5}(1 + \beta_{\mathcal{F}})$ as we just exploit $\mathbf{T}_{\mathbf{G}}$ as the secret key. Hence, $\sigma_k = \|\widetilde{\mathbf{E}}_{y,f_1,\cdots,f_k}\| \cdot \omega(\sqrt{\log((k+1)m)}) = \omega(\beta_{\mathcal{F}} \cdot \sqrt{\log m})$ for all $k \in [\eta]$.
  3. Compared with σ_k of the first approach, σ_k's of the second approach are essentially smaller. Therefore, in order for both trapdoor algorithms and the security to work, we should set $\sigma_1 = \omega(\beta_{\mathcal{F}} \cdot \sqrt{\log m})$ and choose $\beta_{\mathcal{F}} > \|\widetilde{\mathbf{T}_{\mathbf{A}}}\| = \sqrt{n \log m}$ and then follow the first approach in setting Gaussian parameters. Recall that, $\beta_{\mathcal{F}} = (\frac{p^d - 1}{p-1} \cdot m)^\tau \cdot 20\sqrt{m} = O((p^{d-1}m)^\tau \sqrt{m})$ by Lemma 6.

– For the correctness: We need Condition (2) to hold, i.e., $(\eta+1)^2 \cdot \sqrt{m} \cdot \omega((\sqrt{m \log m})^\eta) \cdot \beta_{\mathcal{F}}^2 + 2 < \frac{1}{4}(q/\chi_0)$.

Sizes of Keys and Ciphertext. Recall that, throughout this work, we set $m = \Theta(n \log q)$. The public key corresponding d variables consists of $d+1$ matrices of dimension $n \times m$ over $\mathbb{Z}_q$. Then the public key size is $O((d+1) \cdot n^2 \log^2 q)$. The initial secret key is the short trapdoor matrix $\mathbf{T}_{\mathbf{A}}$ of dimension $m \times m$ generated by TrapGen such that $\|\mathbf{T}_{\mathbf{A}}\| \leq O(\sqrt{n \log q})$, then size is $O(n^2 \log^2 q \cdot \log(n \log q))$. The secret key after delegating η functions is the trapdoor matrix $\mathbf{T}_{y,f_1,\cdots,f_\eta}$ of dimension $(\eta+1)m \times (\eta+1)m$ and $\|\mathbf{T}_{y,f_1,\cdots,f_\eta}\| < \sigma_\eta \cdot \sqrt{(\eta+1)m} = \beta_{\mathcal{F}} \cdot \omega((\sqrt{m \log m})^\eta)$ with overwhelming probability by Lemma 1. Therefore its size is $(\eta+1) \cdot n \log q \cdot (O(\log(\beta_{\mathcal{F}}) + \eta \cdot \log(n \log q)))$. The ciphertext is a tuple of $(d+2)$ vectors of in $\mathbb{Z}_q^m$ hence its size is $O((d+2) \cdot n \log^2 q))$.

4.3 LWE-based PE Construction from DFKHE

We define the family of equality functions $\mathcal{F} := \{f_{t^*} : \mathbb{Z}_q^d \to \mathbb{Z}_q | t^* \in \mathbb{Z}_q\}$, where $f_{t^*}(\mathbf{t}) := eq_{t^*}(t_1) + \cdots + eq_{t^*}(t_d)$, $\mathbf{t} = (t_1, \cdots, t_d)$, $eq_{t^*} : \mathbb{Z}_q \to \mathbb{Z}_q$, satisfying that $\forall t \in \mathbb{Z}_q$, $eq_{t^*}(t) =$

$1 \pmod q$ iff $t = t^*$, otherwise $eq_{t^*}(t) = 0 \pmod q$. Then $f_{t^*}(\mathbf{t}) = 0 \pmod q$ iff $eq_{t^*}(t_i) = 0 \pmod q$ if $d < q$, for all $i \in [d]$. By applying the generic framework in Section 3 to DFKHE demonstrated in Subsection 4.2 and modifying the resulting PE, we come up with the LWE-based PE construction $\Psi = \{\mathsf{PE.key}, \mathsf{PE.enc}, \mathsf{PE.pun}, \mathsf{PE.dec}\}$ presented below:

$\underline{\mathsf{PE.key}(1^\lambda)}$: For the input security parameter λ, do the following:

1. Choose $n = n(\lambda)$, $q = q(\lambda)$ prime, and the maximum number of tags $d = d(\lambda)$ per a ciphertext such that $d < q$.
2. Choose $m = \Theta(n \log q)$. The plaintext space is $\mathcal{M} := \{0,1\}^m$, $\mathcal{T} := \mathbb{Z}_q$. Additionally, let χ be a χ_0–bounded noise distribution (i.e, its support belongs to $[-\chi_0, \chi_0]$ for which the $(n, 2m, q, \chi)$–DLWE is hard. Set $\sigma = \omega(\beta_{\mathcal{F}} \cdot \sqrt{\log m})$.
3. Sample $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$, $\mathbf{U}, \mathbf{B}_1, \cdots, \mathbf{B}_d \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.
4. Output $pk = \{\mathbf{A}, \mathbf{B}_1, \cdots \mathbf{B}_d, \mathbf{U}\}$ and $sk_0 = \{\mathbf{T_A}\}$.

$\underline{\mathsf{PE.enc}(\mu, pk, \{t_1, \cdots, t_d\})}$: For the input consiting of (a message μ, the public key pk and ciphertext tags $(t_1, \cdots, t_d) \in \mathcal{T}^d$), perform the following steps:

1. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_{\mathsf{out}}, \mathbf{e}_{\mathsf{in}} \leftarrow \chi^m$, $\mathbf{S}_1, \cdots, \mathbf{S}_d \xleftarrow{\$} \{-1,1\}^{m \times m}$.
2. Compute $\mathbf{e} \leftarrow (\mathbf{I}_m | \mathbf{S}_1 | \cdots | \mathbf{S}_d)^T \mathbf{e}_{\mathsf{in}} = (\mathbf{e}_{\mathsf{in}}^T, \mathbf{e}_1^T, \cdots, \mathbf{e}_d^T)^T$.
3. Form $\mathbf{H} \leftarrow [\mathbf{A} | t_1\mathbf{G} + \mathbf{B}_1 | \cdots | t_d\mathbf{G} + \mathbf{B}_d]$ and compute $\mathbf{c} = \mathbf{H}^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^{(d+1)m}$, $\mathbf{c} = [\mathbf{c}_{\mathsf{in}} | \mathbf{c}_1 | \cdots | \mathbf{c}_d]$, where $\mathbf{c}_{\mathsf{in}} = \mathbf{A}^T\mathbf{s} + \mathbf{e}_{\mathsf{in}}$ and $\mathbf{c}_i = (t_i\mathbf{G} + \mathbf{B}_i)^T\mathbf{s} + \mathbf{e}_i$ for $i \in [d]$.
4. Compute $\mathbf{c}_{\mathsf{out}} \leftarrow \mathbf{U}^T\mathbf{s} + \mathbf{e}_{\mathsf{out}} + \mu\lceil \frac{q}{2} \rceil$, output $(ct = (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_1, \cdots, \mathbf{c}_d, \mathbf{c}_{\mathsf{out}}), (t_1, \cdots, t_d))$.

$\underline{\mathsf{PE.pun}(sk_{\eta-1}, t_\eta^*)}$: For the input (a puncture key $sk_{\eta-1}$ and a punctured tag $t_\eta^* \in \mathcal{T}$), do:

1. Evaluate $\mathbf{B}_{eq_\eta} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f_{t_\eta^*}, (\mathbf{B}_k)_{k=1}^d)$.
2. Compute $\mathbf{E}_{eq_\eta} \leftarrow \mathsf{ExtBasisLeft}([\mathbf{A} | \mathbf{B}_{eq_1} | \cdots | \mathbf{B}_{eq_{\eta-1}} | \mathbf{B}_{eq_\eta}], \mathbf{T}_{eq_{\eta-1}})$.
3. $\mathbf{T}_{eq_\eta} \leftarrow \mathsf{RandBasis}([\mathbf{A} | \mathbf{B}_{eq_1} | \cdots | \mathbf{B}_{eq_{\eta-1}} | \mathbf{B}_{eq_\eta}], \mathbf{E}_{eq_\eta}, \sigma_\eta)$.
4. Output $sk_\eta := (\mathbf{T}_{eq_\eta}, (t_1^*, \cdots, t_\eta^*), (\mathbf{B}_{eq_1}, \cdots, \mathbf{B}_{eq_\eta}))$.

$\underline{\mathsf{PE.dec}(ct, \mathbf{t}, (sk_\eta, \{t_1^*, \cdots, t_\eta^*\}))}$: For the input (a ciphertext $ct = (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_1, \cdots, \mathbf{c}_d, \mathbf{c}_{\mathsf{out}})$, the associated tags $\mathbf{t} = (t_1, \cdots, t_d)$, a puncture key sk_η and the associated punctured tags $\{t_1^*, \cdots, t_\eta^*\} \subset \mathcal{T}$), execute the following steps:

1. If there exists $j \in [\eta]$ such that $f_{t_j^*}(\mathbf{t}) \neq 0$, then output $\perp$. Otherwise, go to Step 2.
2. Parse $sk_\eta := (\mathbf{T}_{eq_\eta}, (t_1^*, \cdots, t_\eta^*), (\mathbf{B}_{eq_1}, \cdots, \mathbf{B}_{eq_\eta}))$.
3. Sample $\mathbf{R} \leftarrow \mathsf{SampleD}([\mathbf{A} | \mathbf{B}_{eq_1} | \cdots | \mathbf{B}_{eq_\eta}], \mathbf{T}_{eq_\eta}, \mathbf{U}, \sigma_\eta)$.
4. Evaluate $\mathbf{c}_{eq_j} \leftarrow \mathsf{Eval}_{\mathsf{ct}}(f_{t_j^*}, ((t_k, \mathbf{B}_k, \mathbf{c}_k))_{k=1}^d)$, for $j \in [\eta]$.
5. Compute $\bar{\mu} = (\bar{\mu}_1, \cdots, \bar{\mu}_m) \leftarrow \mathbf{c}_{\mathsf{out}} - \mathbf{R}^T(\mathbf{c}_{\mathsf{in}} | \mathbf{c}_{eq_1} | \cdots | \mathbf{c}_{eq_\eta})$.
6. For $\ell \in [m]$, if $|\bar{\mu}_\ell| < q/4$ then output $\mu_\ell = 0$; otherwise, output $\mu_\ell = 1$.

We remark that all analysis done for the LWE-based DFKHE in Subsection 4.2 can perfectly applied to our LWE-based PE. Therefore, we do not mention the analysis again in this section. For completeness, we only state two main theorems as below.

Theorem 4 (Correctness of Ψ). *The proposed* PE Ψ *is correct if* $(\eta+1)^2 \cdot m^{1+\frac{\eta}{2}} \cdot \omega((\sqrt{\log m})^{\eta+1}) \cdot \beta_{\mathcal{F}}^2 + 2 < \frac{1}{4}(q/\chi_0)$, *assumming that* $t_j^* \neq t_k$ *for all* $(j,k) \in [\eta] \times [d]$.

Theorem 5 (IND-sPUN-CPA). *The proposed PE* Ψ *scheme is IND-sPUN-CPA thanks to the IND-sVAR-CPA of the underlying DFKHE* Π.

5 Discussion on Unbounded Number of Ciphertext Tags

The idea of [8] might help us to extend the LWE-based DFKHE construction from Subsection 4.2 (resp., PE from Subsection 4.3) to a variant that supports arbitrary number of variables (resp., ciphertext tags). We call this variant *unDFKHE*. Although, the original idea of [8] is applied to ABE with attributes belonging to $\{0,1\}$ using the XOR operation, we believe that it might be adapted to work well with our DFKHE with variables and punctures over $\mathbb{Z}_q$ using the addition modulo q (denoted $\oplus_q$).

In $\mathsf{unDFKHE}$, the maximum number of ciphertext tags d is not fixed in advance. Then, in the key generation algorithm, we cannot generate $\mathbf{B}_1, \cdots, \mathbf{B}_d$ and give them to the public. In order to solve this issue, we utilize a family of pseudorandom functions PRF=($\mathsf{PRF.Gen}$, $\mathsf{PRF.Eval}$), where $\mathsf{PRF.Gen}(1^\lambda)$ takes as input a security parameter λ and outputs a seed $\mathbf{s} \in \mathbb{Z}_q^\ell$ of length $\ell = \ell(\lambda)$ (which depends on λ) and $\mathsf{PRF.Eval}(\mathbf{s}, \mathbf{x})$ takes as input a seed $\mathbf{s} \in \mathbb{Z}_q^\ell$ and a variable $\mathbf{x} \in \mathbb{Z}_q^*$ of *arbitrary length* and returns an element in $\mathbb{Z}_q$. The family of pseudorandom functions helps us to stretch a variable of fixed length ℓ to one of arbitrary length d as follows. In $\mathsf{unDFKHE.KGen}$, for a variable $\mathbf{t}$ of length $d = |\mathbf{t}|$, instead of $\mathbf{B}_1, \cdots, \mathbf{B}_d$, we generate $\overline{\mathbf{B}}_1, \cdots, \overline{\mathbf{B}}_\ell$ and use them to produce $\mathbf{B}_1, \cdots, \mathbf{B}_d$ later. This can be done by running $\mathsf{Eval}_{\mathsf{pk}}(\mathsf{PRF.Eval}(\cdot, i), (\overline{\mathbf{B}}_k)_{k=1}^{\ell})$, for $i \in [d]$, where $\mathsf{PRF.Eval}(\cdot, i)$ acts as a function that can be evaluated by $\mathsf{Eval}_{\mathsf{pk}}$. Accordingly, any function $f \in \mathcal{F}$ will also be transformed to f_Δ defined by $f_\Delta(\mathbf{t}) := f(\mathbf{t} \oplus_q \Delta_{\leq d})$ before joining to any computation later on. Here $\Delta_i := \mathsf{PRF.Eval}(\mathbf{s}, i)$ for $i \in [d]$, $\Delta_{\leq d} = (\Delta_1, \cdots, \Delta_d)$. Also remark that, $f_\Delta(\mathbf{t} \oplus_q (q_{\leq d} - \Delta_{\leq d})) = f(\mathbf{t})$, where $q_{\leq d} = (q, \cdots, q) \in \mathbb{Z}^d$. Therefore, in $\mathsf{unDFKHE.KHom}$, $\mathbf{B}_f \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f_\Delta, (\mathbf{B}_k)_{k=1}^{d})$.

Actually, there are a lot of work left to be done. Due to space limitation, we leave details of this section for the full version of this paper.

6 Conclusion and Future Works

In this paper, we show puncturable encryption can be constructed from the so-called delegatable fully key-homomorphic encryption. From the framework, we instantiate our puncturable encryption construction using LWE. Our puncturable encryption enjoys the selective indistinguishability under chosen plaintext attacks, which can be converted into adaptive indistinguishability under chosen ciphertext attacks using well-known standard techniques. For future works, there are few investigation directions worth pursuing such as design of: (i) puncturable lattice-based ABE as in [24], (ii) efficient puncturable forward-secure encryption schemes as proposed in [18] or (iii) puncturable encryption schemes, whose puncture key size is constant or puncturable ecnryption schemes support unlimited number of punctures.

Acknowledgment. We thank Sherman S.M. Chow and anonymous reviewers for their insightful comments which improve the content and presentation of the manuscript. This work is supported by the Australian Research Council Linkage Project LP190100984. Huy Quoc Le has been sponsored by a CSIRO Data61 PhD Scholarship and CSIRO Data61 Top-up Scholarship. Josef Pieprzyk has been supported by the Australian ARC grant DP180102199 and Polish NCN grant 2018/31/B/ST6/03003.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 553–572. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28

2. Alkim, E., Bos, J.W., Ducas, L., et al.: FrodoKEM: Learning with Errors Key Encapsulation (Algorithm Specifications And Supporting Documentation, version 25 March, 2020 (2020)), Available from: https://frodokem.org/ Accessed on 08 July, 2020.
3. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings. pp. 75–86 (2009). https://doi.org/10.4230/LIPIcs.STACS.2009.1832
4. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. Journal of Cryptography **24(4**, 659–693 (2011)
5. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits. vol. 8441 (10 2014). https://doi.org/10.1007/978-3-642-55220-5_30
6. Boneh, D., Kim, S., Montgomery, H.: Private Puncturable PRFs from Standard Lattice Assumptions. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 415–445. Springer International Publishing, Cham (2017)
7. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) Theory of Cryptography. pp. 253–273. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
8. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: Unbounded Attributes and Semi-adaptive Security. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 363–384. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. pp. 207–222. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
10. Canetti, R., Raghuraman, S., Richelson, S., Vaikuntanathan, V.: Chosen-Ciphertext Secure Fully Homomorphic Encryption. In: Fehr, S. (ed.) Public-Key Cryptography – PKC 2017. pp. 213–240. Springer Berlin Heidelberg, Berlin, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_8
11. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 523–552. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
12. Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking Cryptographic Capabilities. In: Wichs, D., Mansour, Y. (eds.) STOC 2016: Proceedings of the forty-eighth annual ACM symposium on Theory of Computing. pp. 1115–1127. Cambridge, MA,USA (2016). https://doi.org/10.1145/2897518.2897651
13. Delerablée, C.: Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. In: Kurosawa, K. (ed.) Advances in Cryptology – ASIACRYPT 2007. pp. 200–215. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
14. Derler, D., Gellert, K., Jager, T., Slamanig, D., Striecks, C.: Bloom Filter Encryption and Applications to Efficient Forward-Secret 0-RTT Key Exchange. Cryptology ePrint Archive, Report 2018/199 (2018), https://eprint.iacr.org/2018/199
15. Derler, D., Jager, T., Slamanig, D., Striecks, C.: Bloom Filter Encryption and Applications to Efficient Forward-Secret 0-RTT Key Exchange. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 425–455. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_14
16. Derler, D., Krenn, S., Lorünser, T., Ramacher, S., Slamanig, D., Striecks, C.: Revisiting Proxy Reencryption: Forward Secrecy, Improved Security, and Applications. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography – PKC 2018. pp. 219–250. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_8
17. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432 (2008), https://eprint.iacr.org/2007/432
18. Green, M.D., Miers, I.: Forward Secure Asynchronous Messaging from Puncturable Encryption. In: 2015 IEEE Symposium on Security and Privacy. pp. 305–320 (May 2015). https://doi.org/10.1109/SP.2015.26
19. Günther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89. pp. 29–37. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
20. Günther, F., Hale, B., Jager, T., Lauer, S.: 0-RTT Key Exchange with Full Forward Secrecy. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 519–548. Springer International Publishing, Cham (2017)
21. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography. pp. 581–600. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
22. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 700–718. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

23. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: 45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings. pp. 372–381 (2004). https://doi.org/10.1109/FOCS.2004.72
24. Phuong, T.V.X., Ning, R., Xin, C., Wu, H.: Puncturable attribute-based encryption for secure data delivery in internet of things. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. pp. 1511–1519 (April 2018). https://doi.org/10.1109/INFOCOM.2018.8485909
25. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93 (2005). https://doi.org/10.1145/1060590.1060603
26. Sun, S., Sakzad, A., Steinfeld, R., Liu, J., Gu, D.: Public-Key Puncturable Encryption: Modular and Compact Constructions. In: Proceedings of the 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Part I. pp. 309–338 (2020)