Predicate Encryption from Lattices: Enhanced Compactness and Refined Functionality

Yuejun Wang[1], Baocang Wang[1], Qiqi Lai[2], and Huaxiong Wang[3]

[1] Xidian University
yuejun.w@stu.xidian.edu.cn, bcwang@xidian.edu.cn
[2] Shaanxi Normal University
laiqq@snnu.edu.cn
[3] Nanyang Technological University
hxwang@ntu.edu.sg

Abstract. In this work, we explore the field of lattice-based Predicate Encryption (PE), with a focus on enhancing compactness and refining functionality.
First, we present a more compact bounded collusion predicate encryption scheme compared to previous constructions, significantly reducing both the per-unit expansion and fixed overhead, while maintaining an optimal linear blow-up proportional to Q.
Next, we propose a Predicate Inner Product Functional Encryption (P-IPFE) scheme based on our constructed predicate encryption scheme. P-IPFE preserves the attribute-hiding property while enabling decryption to reveal only the inner product between the key and message vectors, rather than the entire message as in traditional PE. Our P-IPFE scheme also achieves bounded collusion resistance while inheriting the linear compactness optimized in the underlying PE scheme. Additionally, it supports any polynomial-sized and bounded-depth circuits, thereby extending beyond the inner-product predicate class in prior works.
Furthermore, all the proposed schemes achieve selective fully attribute-hiding security in the simulation-based model, therefore, can further attain semi-adaptive security by adopting existing upgrading techniques.

1 Introduction

Functional Encryption (FE) [O'N10, BSW11] is a groundbreaking cryptographic paradigm that allows fine-grained control over how encrypted data is accessed, moving beyond the traditional "all-or-nothing" approach. In a FE scheme, given an encryption of input x, a functional secret key associated with a function f can decrypt the ciphertext to reveal the functional output $f(x)$, while revealing nothing else about x. This contrasts with traditional encryption schemes, where successful decryption yields the entire plaintext. Importantly, FE forms the foundation for constructing advanced cryptographic primitives, such as reusable garbled circuit schemes [GKP+13] and indistinguishability obfuscation (iO) [AJ15, BV15].

Within the broad framework of FE, Predicate Encryption (PE) [KSW08] stands out as a powerful and practical special case, particularly useful in real-world applications. From the aspect of correctness requirement, PE operates similarly to Attribute-Based Encryption (ABE). Intuitively, the underlying message is revealed only when the attribute associated with the ciphertext satisfies the predicate function tied to the secret decryption key. Moreover, compared with traditional ABE which commonly assume public attributes, PE offers enhanced privacy by keeping attributes hidden, making it a more suitable solution for scenarios that requires attribute confidentiality. Consequently, the security definition of PE is also more complex due to its additional attribute privacy.

The basic security guarantee for attributes in predicate encryption is weak attribute-hiding, which ensures that attributes remain hidden as long as the adversary cannot decrypt corresponding ciphertexts. Specifically, in the security game, the adversary is restricted to only querying secret keys for predicates f_i such that $f_i(\mathbf{x}^*) =$ false, where $\mathbf{x}^*$ is the challenge attribute. In contrast, fully attribute-hiding security imposes no such limitation, allowing the adversary to obtain any secret key, including those for predicates f_i where

$f_i(\mathbf{x}^*) =$ true. This stronger security notion thus guarantees that no information about the attribute is revealed, regardless of whether the decryption succeeds or fails.

A predicate encryption scheme with fully attribute-hiding property can further imply FE, as shown in the literature [Agr17, LLW21]. For instance, to construct a regular FE for the boolean function class $\mathcal{F} : \mathcal{X} \to \{0, 1\}$, one can rely on a PE scheme that supports the predicate class $\mathcal{F}$ and attribute space $\mathcal{X}$. In particular, the secret key for a function $f \in \mathcal{F}$ in FE scheme is set to the policy-related key for f in the underlying PE scheme. The FE ciphertext for an input $x \in \mathcal{X}$ is accordingly computed by running the PE encryption algorithm for the attribute x and an arbitrary message bit μ. The functionality value $f(x)$ can hence be determined by checking whether the decryption of the underlying PE, with the FE secret key and ciphertext as inputs, succeeds or fails. Additionally, both the supporting function class and the achieved security level are inherited in the resulting FE scheme.

On the other hand, the fully attribute-hiding functionality directly aligns with the concept of computation hiding discussed in [BSW11]. While, as proposed in [BSW11], it is more appropriate to consider a simulation-based security definition for FE schemes whose functionality inherently provides computation hiding. Therefore, most recent constructions [Agr17, DOT18, LLW21, DDM+23], including weak attribute-hiding schemes [GVW15, BTVW17, Wee17], adopt the simulation-based security model.

In most practical application scenarios, an adversary typically has only limited computational power and can only collude with a limited number of parties. Given this, bounded collusion-resistance aligns better with realistic security requirements and efficiency needs. More specifically, it ensures security in the premise that the adversary obtains at most a-prior bounded Q secret keys. Bounded collusion FE [GVW12, Agr17, AR17, AV19, LLW21] has been extensively studied in the past decade following various technical approaches. In [LLW21], Lai et al. made a significant progress by proposing a FE scheme achieving an additional $O(Q)$ blow-up in ciphertext and public key size. In addition, the security is ensured against up to Q authorized key (1-key) queries and any polynomial number of unauthorized key (0-key) queries. As demonstrated in [AGVW13], the ciphertext size of FE schemes grows at least linearly with the collusion bound Q. Therefore, the construction in [LLW21] achieves an optimal blow-up, considering this lower bound. Nevertheless, as we will explain more clearly in the technical background, a noticeable portion of the per-unit expansion overhead in both ciphertexts and keys can be avoided. In other words, further optimization in compactness is achievable by adopting a more efficient construction approach. Although this may seem theoretical, such improvements in compactness are a necessary step toward practical applications.

To further refine the functionality of PE, each secret key can be tied to a key vector, in addition to the predicate, allowing for more fine-grained operations. Furthermore, each ciphertext is linked to both a message vector and an attribute. The decryption then outputs an inner product between the key vector and the message vector, provided that the predicate-attribute pair matches. This extension not only enriches the applicability of PE but also offers a step toward a Predicate Inner Product Functional Encryption (P-IPFE), first introduced in [DDM+23]. As a practical class of FE, P-IPFE enables more expressive access control while ensuring attribute privacy. Additionally, P-IPFE can be viewed as an Attribute-based IPFE (AB-IPFE) scheme [ACGU20, LLW21] with additional attribute-hiding property, which is critical for sensitive applications such as medical data management or voting systems.

In [DDM+23], Dowerah et al. presented pairing-based unbounded (non-)zero predicate IPFE schemes that satisfy fully attribute-hiding. Specifically, the proposed unbounded non-zero predicate IPFE scheme achieves strong attribute-hiding in the simulation-based model, while the supported predicate classes are restricted to unbounded non-zero inner-product predicates, which are inherently linear. Although more complex predicates can be supported by representing them as inner-product computations, the associated overhead may become prohibitive. In other words, the inner-product predicate restricts their applicability for scenarios requiring non-linear evaluations in some extent. Therefore, constructing a predicate IPFE scheme that supports a more general class of predicate function, beyond linear functions, remains an interesting open question and is one of the focuses of this work.

1.1 Our Contributions

In this work, we make several contributions to the field of Predicate Encryption:

- *More Compact Predicate Encryption Scheme.* We propose a bounded collusion predicate encryption scheme for any polynomial-sized, bounded-depth circuits. The construction significantly reduces the ciphertext and key size while preserving fully attribute-hiding security. More specifically, by adopting a more efficient approach to achieving attribute-hiding, our scheme retains an *optimal* additional linear blow-up with respect to the collusion bound Q and more importantly, optimizes both the per-unit expansion and fixed overhead. We therefore offer a more compact design compared to previous constructions. Comparison with prior schemes are provided in Table 1.
- *Predicate IPFE Scheme for General Predicate Function.* We present a predicate IPFE that allows any polynomial sized, bounded-depth circuits, extending beyond the inner-product predicate supported in prior constructions. Our scheme allows up to Q 1-keys and achieve fully attribute-hiding security in the simulation-based security model. Through the techniques we developed for achieving attribute-hiding and bounded collusion resistance, our constructed P-IPFE supports more refined functionality while inheriting the linear compactness optimized in the PE scheme.

All the proposed schemes are formally proven to be selectively secure based on learning with errors (LWE) assumptions in the standard model. Furthermore, they can be further upgraded to achieve semi-adaptive security by following the approaches in [GKW16, BV16]. Particularly, our construction strategy is compatible with the light-weight upgrading method in [BV16], enabling a more efficient semi-adaptive construction compared with previous schemes. We refer the readers to Appendix E for semi-adaptive constructions and further comparisons.

Succinctness is also achieved in all constructions, i.e., the ciphertext size is independent of the circuit size. Furthermore, our proposed PE construction can naturally be used to construct FE for general circuits, following the approach in [Agr17, LLW21]. Starting with the compact PE scheme presented in this work, the resulting FE scheme preserves the succinctness and achieves improved efficiency.

	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{ct}\rvert$
[Agr17]	$(O(Q^2)+\ell\cdot\lvert\mathsf{hct}\rvert)\cdot\lvert\mathbb{Z}_q^{n\times m}\rvert$ $+(O(Q^2)\cdot\lvert\mathsf{hct}\rvert+\lvert\mathsf{hsk}\rvert)\cdot\lvert\mathbb{Z}_q^{n\times m}\rvert$	$\ell\cdot\lvert\mathsf{hct}\rvert+((O(Q^2)+\ell\cdot\lvert\mathsf{hct}\rvert)\cdot\lvert\mathbb{Z}_q^{m}\rvert$ $+O(Q^2)\cdot\lvert\mathsf{hct}\rvert+(O(Q^2)\cdot\lvert\mathsf{hct}\rvert+\lvert\mathsf{hsk}\rvert)\cdot\lvert\mathbb{Z}_q^{m}\rvert$
[LLW21]	$(O(Q)+\ell\cdot\lvert\mathsf{hct}\rvert)\cdot\lvert\mathbb{Z}_q^{n\times m}\rvert$ $+(O(Q)\cdot\lvert\mathsf{hct}\rvert+\lvert\mathsf{hsk}\rvert)\cdot\lvert\mathbb{Z}_q^{n\times m}\rvert$	$\ell\cdot\lvert\mathsf{hct}\rvert+((O(Q)+\ell\cdot\lvert\mathsf{hct}\rvert)\cdot\lvert\mathbb{Z}_q^{m}\rvert$ $+O(Q)\cdot\lvert\mathsf{hct}\rvert+(O(Q)\cdot\lvert\mathsf{hct}\rvert+\lvert\mathsf{hsk}\rvert)\cdot\lvert\mathbb{Z}_q^{m}\rvert$
Ours	$(O(Q)+\ell\cdot\lvert\mathsf{hct}\rvert)\cdot\lvert\mathbb{Z}_q^{n\times m}\rvert$	$\ell\cdot\lvert\mathsf{hct}\rvert+((O(Q)+\ell\cdot\lvert\mathsf{hct}\rvert)\cdot\lvert\mathbb{Z}_q^{m}\rvert$

Table 1. Comparison with previous Q-collusion resistant PE constructions. Specifically, we compared our selective PE with a selective PE [LLW21] and a very-selective one [Agr17]. We denote the bit-length of attribute as ℓ, the size of homomorphic encryption ciphertext (for 1-bit) and secret key by $\lvert\mathsf{hct}\rvert$ and $\lvert\mathsf{hsk}\rvert$, respectively. The size of an element in $\mathbb{Z}_q^{n\times m}$ (resp. $\mathbb{Z}_q^{m}$) is denoted by $\lvert\mathbb{Z}_q^{n\times m}\rvert$ (resp. $\lvert\mathbb{Z}_q^{m}\rvert$).

1.2 Technical Background

In this section, we review several crucial techniques for building Predicate Encryption from lattices that have been developed in the past decade, and analyze which partial overhead in previous schemes can potentially be further reduced.

In many cases, fully homomorphic encryption (FHE) is a powerful tool for achieving attribute-hiding while also enabling homomorphic evaluation on encrypted attribute encodings, therefore effectively upgrading ABE to PE. Intuitively, one can encrypt an attribute before generate its encoding. Similar to ABE, the decryption process of PE must determine whether the attribute x satisfies the predicate f, i.e., whether $f(x) = 0$. However, the homomorphic evaluation produces an encoding of encrypted $f(x)$. This thus requires a process known as "eval-then-dec", where the encrypted attribute encodings are first evaluated homomorphically, followed by decryption, which yields the encoding for plaintext result $f(x)$, as desired. The "eval" process is essentially similar among prior constructions, while the approaches to implementing "dec" vary. Before introducing the different approaches to achieving "dec", we first review the structure of attribute encoding and its homomorphism properties.

Attribute Encodings and Homomorphic Evaluations. Given a public matrix $\mathbf{A}_{\mathsf{attr}}$, the encoding for an attribute $\mathbf{x}$ is computed as $\underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G})}$, where $\mathbf{s}$ is a chosen LWE secret and the underline denotes the noise term. For a Boolean circuit f, we can compute a matrix $\mathbf{H}_f$. Another matrix $\mathbf{H}_{f,\mathbf{x}}$ can also be computed when given $\mathbf{x}$ additionally. Moreover, the following key equations are hold.

$$(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} = \mathbf{A}_{\mathsf{attr}}\mathbf{H}_f + f(x) \cdot \mathbf{G}$$
$$\underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G})} \cdot \mathbf{H}_{f,\mathbf{x}} = \underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}}\mathbf{H}_f + f(x) \cdot \mathbf{G})}$$

When we use the GSW FHE [GSW13] to hide the attribute $\mathbf{x}$, it can be first encrypted to $\Psi_\mathbf{x} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}_{\mathsf{HE}}^\top\mathbf{A} + \mathbf{e}^\top \end{pmatrix} \cdot \mathbf{R}_\mathbf{x} + \mathbf{x} \otimes \mathbf{G}$, where $\mathbf{r}_{\mathsf{HE}}^\top = (\mathbf{s}_{\mathsf{HE}}^\top, -1)$ is the corresponding secret key and $\mathbf{R}_\mathbf{x}$ is the encryption randomness. Then, the encoding for the encrypted attribute $\Psi_\mathbf{x}$ will be formed as

$$\underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}} + \Psi_\mathbf{x}^\top \otimes \mathbf{G})}$$

Similarly, we can compute matrices $\mathbf{H}_{\mathsf{HEval}_f}$ and $\mathbf{H}_{\mathsf{HEval}_f,\Psi_\mathbf{x}}$, where HEval_f describes the circuit of the homomorphic evaluation related to f. Based on the encoding homomorphism for matrix-valued circuit as proposed in [BTVW17], we have

$$\underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}} + \Psi_\mathbf{x}^\top \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathsf{HEval}_f,\Psi_\mathbf{x}} = \underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}}\mathbf{H}_{\mathsf{HEval}_f} + \Psi_{f(\mathbf{x})})}$$

Different Approaches to Decryting Encoding. To decrypt the resulting encoding, a natural approach is to apply an operation analogous to the decryption process in FHE, using the corresponding FHE secret key $\mathbf{r}_{\mathsf{HE}}$. More precisely, we can first generate an additional encoding for the FHE secret key $\mathbf{r}_{\mathsf{HE}}$. Then, the evaluation procedure is defined as an FHE evaluation followed by FHE decryption, applied between the secret key and the homomorphically evaluated ciphertext, as described below.

$$\begin{aligned} &\underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}} + (\Psi_\mathbf{x}|\mathbf{r}_{\mathsf{HE}})^\top \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathsf{HEval}_f \circ \mathsf{HDec},(\Psi_\mathbf{x},\mathbf{r}_{\mathsf{HE}})} \\ =\ &\underline{\mathbf{s}^\top(\mathbf{A}_{\mathsf{attr}}\mathbf{H}_{\mathsf{HEval}_f \circ \mathsf{HDec}} + f(\mathbf{x}) \cdot \mathbf{G})} \end{aligned}$$

According to the aforementioned properties, the evaluator must know $\mathbf{r}_{\mathsf{HE}}$ for homomorphic evaluations. Obviously, including $\mathbf{r}_{\mathsf{HE}}$ in the ciphertext would expose the attribute entirely. On the positive side, FHE decryption essentially reduces to an inner product between the secret key and the ciphertext, followed by a threshold operation. Furthermore, as pointed out in [GVW15], the lack of secret key itself does not hinder the evaluation of the inner product between the encoding of secret key and ciphertext. Indeed, the ciphertext is sufficient for this encoding evaluation. Besides, a "lazy-OR" trick can be applied to bridge the gap between threshold inner products and simple inner products. More precisely, a secret key is associated with a bunch of predicate functions, each corresponding to a possible decryption noise value after the "eval-then-dec" process.

Another approach is to enable automatic decryption. Specifically, due to the structural similarity between encodings and homomorphic evaluations in both GSW FHE scheme [GSW13] and the ABE scheme [BGG+14], the same secret can be used for both encryption and encoding. Such a technique is introduced in [BTVW17] and named by dual-use. By adopting the FHE secret key $\mathbf{r}_{\mathsf{HE}}$ as the randomness for encrypted attribute encoding, the homomorphic evaluation output would automatically align with the decryption process as follows.

$$\begin{aligned} &\underline{\mathbf{r}_{\mathsf{HE}}^\top(\mathbf{A}_{\mathsf{attr}} + \Psi_\mathbf{x}^\top \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathsf{HEval}_f,\Psi_\mathbf{x}} \\ =\ &\underline{\mathbf{r}_{\mathsf{HE}}^\top(\mathbf{A}_{\mathsf{attr}}\mathbf{H}_{\mathsf{HEval}_f})} + \underbrace{\underline{\mathbf{r}_{\mathsf{HE}}^\top \cdot \Psi_{f(\mathbf{x})}}}_{\text{FHE decryption}} \\ =\ &\underline{\mathbf{r}_{\mathsf{HE}}^\top(\mathbf{A}_{\mathsf{attr}}\mathbf{H}_{\mathsf{HEval}_f} + f(\mathbf{x}) \cdot \mathbf{G})} \end{aligned}$$

Efficiency Analysis of Different Decryption Approaches. For weak attribute hiding PE schemes, the aforementioned two decryption approaches bring relatively close efficiency performances. Shortly speaking, the usage of the first approach will additionally bring a private attribute encoding for the FHE secret key in the ciphertext, as well as corresponding encoding matrices in the public key.

For fully attribute-hiding PE schemes, however, we observe that all existing constructions adopt the first approach. Specifically, the scheme [BTVW17], which uses the dual-use technique, only achieves weaker security. We thus turn to analyzing fully attribute-hiding schemes built from the first approach. Technically, in proving security, the challenger needs to simulate 1-key queries. Each 1-key should successfully decrypt the challenge ciphertext by correctly recomputing the one-time pad (OTP). Moreover, as for Q bounded collusion schemes, the adversary can obtain at most Q 1-keys. This requires that the OTPs computed by different 1-keys remain independent from one another. To achieve this, cover-free set technique [GVW12, AV19, LLW21] is leveraged [Agr17, AV19, LLW21] to sample independent subsets $\varDelta \subseteq [N]$ for secret (1-)keys. Accordingly, each ciphertext contains N copies of the payload, with independent public matrices. Intuitively, cover-freeness ensures that each subset contains a unique index that does not appear in any of the other $Q-1$ subsets. Hence, the OTPs computed by different 1-keys can remain independent, as desired. On the other hand, this also leads to a blow-up in both the public key and ciphertext size, proportional to Q. Though, as shown in [AGVW13], such blow-up is unavoidable. Moreover, a novel sampling approach [LLW21] for cover-free sets allows an optimal blow-up with $O(Q)$.

Furthermore, as is pointed out in [Agr17], after applying the "lazy-OR" technique, a secret key would expose the exact FHE decryption noise upon successful decryption, thus posing a security risk. To mitigate this, additional FHE dummy ciphertexts must be introduced to mask the original FHE randomness, becoming part of the public attribute alongside the encrypted attribute. These dummy ciphertexts must also be replicated into N copies (in the Q-bounded collusion scheme), as the FHE noise resulting from each 1-key needs to remain independent.

We observe that beyond the increase from FHE secret key encoding, the FHE dummy ciphertexts and corresponding encoding matrices contribute significantly to the per-unit expansion, which is less than ideal. More importantly, this overhead is directly attributed to the first decryption approach.

1.3 Technical Overview

In this section, we present the high-level idea of constructing succinct bounded collusion-resistant predicate encryption and predicate IPFE scheme. Specifically, the proposed predicate encryption is more compact than previous constructions.

More Compact Predicate Encryption Using Dual-Use Technique. To further reduce the per-unit overhead, we explore constructing a bounded collusion PE scheme using the dual-use technique, following the steps outlined below. We define (q_1, q_0) as the parameters describing the admissible number of 1-key and 0-key queries, respectively.

$$\begin{aligned}
&(0, \mathsf{poly}) \text{ selective PE [BTVW17]}\\
&\qquad\downarrow \text{ two-stage sampling technique [LLW21]}\\
&(1, \mathsf{poly}) \text{ selective PE}\\
&\qquad\downarrow \text{ cover-free set}\\
&(Q, \mathsf{poly}) \text{ selective PE}\\
&\qquad\downarrow \text{ light-weight upgrading [BV16]}\\
&(Q, \mathsf{poly}) \text{ semi-adaptive PE}
\end{aligned}$$

Roughly speaking, the novel two-stage sampling technique, first proposed in [LLW21], allows answering 1-key queries without requiring the adversary to submit 1-key queries before the public matrices been generated. In other words, this enables us to upgrade the weak attribute-hiding PE scheme in [BTVW17] to fully attribute-hiding one against a single 1-key query. To further achieve bounded collusion-resistance, we can rely on the improved cover-freeness property [LLW21] to sample independent subset $\Delta \subseteq [N]$ for each secret key. Specifically, each ciphertext includes $N = O(Q)$ independent copies of the payload. During the proof, we can pre-sample Q subsets (or $2Q$ subsets, depending on whether the adversary queried for 1-key before challenge) for the forthcoming 1-key queries, either in the pre-challenge or post-challenge phase. Consequently, decryption correctness can be promised by carefully generating secret shares for the message encoding. Due to the usage of dual-use technique, the predicate-related matrix for the 1-key is connected not only to the randomness used for programming encoding matrix, but also to the encryption randomness $\mathbf{R}_{\mathbf{x}}$ of the FHE ciphertext Ψ_x. Therefore, we must carefully remove all dependencies on $\mathbf{R}_{\mathbf{x}}$ except for Ψ_x step by step. Finally, we show that the attribute $\mathbf{x}$ remains perfectly hidden by Leftover Hash Lemma (LHL).

Typically, our construction approach directly achieves fully attribute-hiding bounded collusion PE, getting rid of building partially-hiding PE (PH-PE) as an intermediate step. We observe that the process of upgrading from PH-PE to PE inherently requires the use of the "lazy-OR" technique. To explain further, each ciphertext in the PH-PE scheme is associated with both a public and a private attribute. In order to construct PE from PH-PE, the attribute is first encrypted using fully FHE and then set as the public attribute of the PH-PE scheme. The corresponding FHE secret key is hidden by being set as the private attribute. However, the PH-PE schemes in [GVW15, Agr17, LLW21] only support inner product predicates over private attributes. Moreover, the decryption of FHE involves a threshold inner product, rather than a simple inner product. As a result, to implement the "eval-then-dec" procedure using the PH-PE scheme, FHE decryption is split into an inner product (between the ciphertext and the secret key), followed by noise checking. This noise checking process thus requires "lazy-OR".

In contrast, in our construction, we bypass the "lazy-OR" technique, saving the overhead associated with encoding FHE secret keys and reducing the sizes of the public key, secret key, and ciphertexts. Upon upgrading to the bounded collusion setting using cover-freeness, the blow-up overhead from dummy FHE ciphertexts is further eliminated. Consequently, the proposed construction results in more compact keys and ciphertexts, both in terms of per-unit expansion and fixed overhead, making our scheme more efficient in practical scenarios.

Regarding to Attacks in [Agr17]. In the paper [Agr17], Agrawal pointed out two distinct 1-key attacks against the previous predicate encryption scheme [GVW15]. Intuitively, the first attack utilizes a potential linear relationship between the decryption noises obtained using different 1-keys for simple predicate functions. Such linear correlation enables the adversary to solve the linear equations and extract the exact noise within the ciphertext. To mitigate this, the noise parameter is carefully chosen to ensure sufficient noise flooding within the decryption process. Furthermore, in the (Q, poly) scheme [Agr17], each decryption noise remains independent for different key due to the use of cover-free set. In our scheme, we also adopt the appropriate parameter choices and leverage cover-free set in the bounded collusion setting, to defend this first attack.

The second attack targets the leakage of the exact FHE decryption noise, which occurs upon successful decryption in [GVW15]. To circumvent it, the dummy FHE ciphertext is used, though at the cost of increased overhead. However, we avoid embedding direct information about the FHE decryption noise in the secret key. As a result, the sensitive noise is effectively masked by other noise terms during decryption. Therefore, our scheme is also resilient to this second attack.

Predicate Inner Product Functional Encryption. Following the construction outline of bounded collusion secure PE, we first construct a predicate IPFE allowing a single pre-challenge 1-key query and then upgrade it to allowing multiple 1-keys.

Constructing Predicate IPFE from PE. We begin with a weaker version of predicate IPFE, namely AB-IPFE [LLW21], where the decryption mechanism and output are identical to P-IPFE, but the attribute is explicitly included in the ciphertexts. An AB-IPFE scheme can be constructed by subtly combining an ABE [BGG+14] with an Inner-Product Functional Encryption (IPFE) scheme [WFL19], as introduced in [LLW21]. Broadly speaking, ABE provides the outer framework, ensuring that any further evaluations on the plaintext vector is allowed only if the attribute satisfies the predicate. Inside this framework, the secret key and payload ciphertext are designed based on the IPFE paradigm. Notably, both the attribute encoding and the attribute-predicate matching take place entirely within the ABE structure. Therefore, we first attempt to build a P-IPFE by replacing the public-attribute ABE with our constructed PE scheme.

To prove the security of this initial construction, we must carefully handle the generation of both secret key and ciphertext. Typically, upon receiving the challenge attribute $\mathbf{x}$, we first encrypt $\mathbf{x}$ as $\Psi_{\mathbf{x}}$ and then encode this encrypted challenge attribute $\Psi_{\mathbf{x}}$ into the attribute-encoding matrix $\mathbf{A}_{\mathsf{attr}}$. This enables us to answer key queries by either the public trapdoor of gadget matrix or two-stage sampling algorithm. For the challenge ciphertext, we rely on the underlying IPFE to show that it can be correctly simulated without requiring the challenge message vector $\mathbf{u}^*$. Note that in this simpler case, where only a *single pre-challenge* 1-key query for $(f, \mathbf{v})$ is allowed, we can then compute a dummy message $\mathbf{u}'$, satisfying the constraint that $\langle \mathbf{u}', \mathbf{v} \rangle = \langle \mathbf{u}^*, \mathbf{v} \rangle$, to replace $\mathbf{u}^*$. In addition, the IPFE scheme [ALS16, ACGU20] with single-challenge security is sufficient for our construction goal. To ensure that the distinguishing advantage of replacing the encrypted message vector $\mathbf{u}^*$ with $\mathbf{u}'$ remains bounded by the security of the IPFE, we need to reduce the indistinguishability of the challenge ciphertext in the P-IPFE scheme to the security of the underlying IPFE.

Obstacle to Proving Security. However, proving the security of this transformation for P-IPFE is more challenging than for AB-IPFE. The dual-use technique for achieving attribute-hiding requires the FHE secret key used for encrypting the attribute to be consistent with the secret randomness used in the attribute encoding, which must also match the randomness used in other LWE instances within the ciphertext. Specifically, in the normal scheme, the attribute is encrypted under the FHE public key $(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)$, where $\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ is also the preamble ciphertext. When attempting to reduce the security to IPFE, we would need to simulate the challenge ciphertext by invoking the IPFE challenger. Upon receiving the IPFE preamble ciphertext, we would set it as the preamble ciphertext in the simulated ciphertext and also use it to encrypt the attribute. The challenge then arises because we must set the attribute encoding matrix using the encrypted attribute during the Setup phase, or we won't be able to answer any secret key queries. However, since the encrypted attribute is only available after the challenge phase, this contradicts the definition of the security experiment, making the proof strategy infeasible.

Double-Use of FHE Randomness. The reason we cannot complete the proof as described above is that the generation of the attribute encoding matrix $\mathbf{A}_{\mathsf{attr}}$, which must occur before the challenge phase, heavily depends on information from the challenge ciphertext—specifically, the preamble ciphertext $\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$. This thus creates a timeline conflict. To resolve this, we attempt removing the reliance of the attribute encoding matrix on the preamble ciphertext in the challenge ciphertext and instead program it using alternative information.

In the normal scheme, each predicate function f defined over the plaintext attribute $\mathbf{x}$ is first encoded into another function $\hat{f}$, defined as $\hat{f} : \Psi_{\mathbf{x}} \mapsto \overline{\Psi}_{f(\mathbf{x})}$, where $\overline{\Psi}_{f(\mathbf{x})}$ describes all but the last row of $\Psi_{f(\mathbf{x})}$. For a FHE ciphertext $\Psi_{\mathbf{x}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \end{pmatrix} \cdot \mathbf{R}_{\mathbf{x}} + \mathbf{x} \otimes \mathbf{G}$, the result after homomorphic evaluation takes the following form:

$$\Psi_{f(\mathbf{x})} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \end{pmatrix} \cdot \mathbf{R}_f + f(x) \cdot \mathbf{G} = \begin{pmatrix} \overline{\Psi}_{f(\mathbf{x})} \\ \underline{\Psi}_{f(\mathbf{x})} \end{pmatrix}$$

The upper part $\overline{\Psi}_{f(\mathbf{x})} = \mathbf{A} \cdot \mathbf{R}_f + f(x) \cdot \overline{\mathbf{G}}$ is independent of the FHE secret key $(\mathbf{s}^\top, -1)$ which encrypts $\Psi_{f(\mathbf{x})}$. This observation allows us to encrypt the attribute $\mathbf{x}$ once more using the same public key $\mathbf{A}$ and encryption randomness $\mathbf{R}_{\mathbf{x}}$, but with another randomness $\mathbf{s}'$. This yields:

$$\Psi'_{\mathbf{x}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}'^\top \mathbf{A} + \mathbf{e}'^\top \end{pmatrix} \cdot \mathbf{R}_{\mathbf{x}} + \mathbf{x} \otimes \mathbf{G}$$

After the same homomorphic evaluation, we find that $\overline{\Psi}'_{f(\mathbf{x})} = \overline{\Psi}_{f(\mathbf{x})}$, meaning the upper part of the ciphertext remains consistent across both encryptions! Thus, we can include both ciphertexts, $\Psi_{\mathbf{x}}$ and $\Psi'_{\mathbf{x}}$, in the construction, using $\Psi'_{\mathbf{x}}$ for encrypted attribute encoding, and retain the dual-use technique's correctness.

In the security reduction to IPFE, $\Psi'_{\mathbf{x}}$ is computed using the fresh randomness $\mathbf{s}', \mathbf{e}'$, and $\mathbf{R}_{\mathbf{x}}$, and is therefore independent of other LWE instances in the ciphertext. It thus enables us to set the attribute encoding matrix during Setup. Additionally, the computation of $\Psi_{\mathbf{x}}$ can be delayed until receiving the IPFE challenge ciphertext. This method resolves the timeline conflict and allows the security reduction to IPFE to proceed successfully.

Towards Bounded Collusion By Extending Dimensions. To construct a bounded collusion-resistant P-IPFE scheme, we draw inspirations from both the (Q, poly) PE framework and the simulation-based secure IPFE scheme proposed in [ALMT20]. Leveraging the cover-free set, as used in the (Q, poly) PE scheme, each key generation involves independent subset sampling, and each ciphertext includes N copies of the payload. In the proof, the challenge ciphertext can be easily simulated without knowing the challenge message $\mathbf{u}^*$ if only pre-challenge 1-key queries are allowed, as a dummy vector $\mathbf{u}'$ can be computed to satisfy $\langle \mathbf{u}', \mathbf{v}_i \rangle = \langle \mathbf{u}^*, \mathbf{v}_i \rangle$ for all ever queried 1-key vectors $\mathbf{v}_i$. However, to accommodate post-challenge 1-key queries, additional programming space is required.

The generic approach introduced in [ALMT20] provides insight into achieving simu-lation-based security by doubling the dimension of both the underlying message and key spaces. Roughly speaking, the scheme and its security proof hinge on the following equations:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \begin{cases} \langle (\mathbf{u}^\top, 0, 0), (\mathbf{v}^\top, 1, r) \rangle & \text{(scheme)} \\ \langle (\mathbf{u}^\top, -r, 1), (\mathbf{v}^\top, 1, r) \rangle & \text{(security proof/pre-challenge)} \\ \langle (\mathbf{u}^\top, -r, 1), (\mathbf{v}^\top, 1, r+\theta) \rangle & \text{(security proof/post-challenge)} \end{cases}$$

When simulating the challenge ciphertext, a similar dummy vector $\mathbf{u}'$ is computed to ensure decryption correctness for all pre-challenge key queries. For post-challenge queries, the randomness r in the encoded key vector additionally absorbs the difference θ between $\langle \mathbf{u}^*, \mathbf{v} \rangle$ and $\langle \mathbf{u}', \mathbf{v} \rangle$.

The need for double dimensions in the simulation-based secure IPFE scheme [ALMT20] arises because each independent randomness (i.e., r) encoded in the key vector requires a corresponding counterpart (i.e., $-r$) in the message vector. Consequently, the message vector must include all sampled randomness, while for each secret key, only two slots are effectively utilized among the expanded dimensions. However, in our (Q, poly) P-IPFE scheme, each additional ciphertext dimension effectively corresponds to N dimensions due to the use of cover-free sets. This enables the counterparts in the message vector to be computed using secret-sharing, where each independent randomness r in the key vector is recomputed by $\sum_{k \in \Delta} r'_k$. As a result, two extra dimensions, combined with secret-sharing techniques, are sufficient to support the security proof. In more detail, we rely on the following equations:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \begin{cases} \langle \sum_{k \in \Delta} (\frac{1}{|\Delta|} \mathbf{u}^\top, 0, 0), (\mathbf{v}^\top, 1, r) \rangle & \text{(scheme)} \\ \langle \sum_{k \in \Delta} (\frac{1}{|\Delta|} \mathbf{u}^\top, -r'_k, 1), (\mathbf{v}^\top, 1, r) \rangle & \text{(security proof/pre-1-key)} \\ \langle \sum_{k \in \Delta} (\frac{1}{|\Delta|} \mathbf{u}^\top, -r'_k, 1), (\mathbf{v}^\top, 1, r+\theta) \rangle & \text{(security proof/post-1-key)} \end{cases}$$

The correctness of the normal scheme is naturally guaranteed as the extended plaintext vector is padded by zeros. Regarding the security proof, we pre-sample Q cover-free subsets and independent randomness $\{r_i\}_{i \in [Q]}$ for the forthcoming Q 1-key queries. Additionally, we generate a secret sharing $\{r'_k\}_{k \in [N]}$ such that $\sum_{\Delta_i} r'_k = r_i$ for each $i \in [Q]$, which is feasible by cover-freeness property. To ensure correct decryption, the key vector is set as $(\mathbf{v}^\top, 1, r_i)$ for pre-challenge 1-key queries, or as $(\mathbf{v}^\top, 1, r_i + \theta_i)$, where θ_i is artificially added to eliminate the difference between the real challenge message and the dummy message vector.

2 Preliminaries

Notation. In this paper, $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{R}$ denote sets of integers, positive integers and real numbers. We use λ to denote the security parameter, which is the implicit input for all algorithms presented in this paper. A function $f(\lambda) > 0$ is *negligible* and is denoted by $\mathsf{negl}(\lambda)$ if for any $c > 0$ and sufficiently large λ, $f(\lambda) < 1/\lambda^c$. A probability is called *overwhelming* if it is $1-\mathsf{negl}(\lambda)$. A function $f(\lambda) > 0$ is *polynomial* if there exists $c \in \mathbb{N}$ and sufficiently large λ, $f(\lambda) \in O(\lambda^c)$. Efficient is used to describe the algorithms that can be performed in probabilistic polynomial time (PPT).

A column vector is denoted by a bold lowercase letter (e.g., $\mathbf{x}$). A matrix is denoted by a bold upper case letter (e.g., $\mathbf{A}$). For a vector $\mathbf{x}$, its Euclidean norm (also known as the ℓ_2 norm) and infinity norm is written as $\|\mathbf{x}\|$ and $\|\mathbf{x}\|_\infty$, respectively. For a matrix $\mathbf{A}$, its i-th column vector is denoted by $\mathbf{a}_i$ and its transposition is denoted by $\mathbf{A}^\top$. We use $\widetilde{\mathbf{A}}$ to denote its Gram-Schmidt orthogonalization. The Euclidean norm and spectral norm of a matrix $\mathbf{A}$ is denoted by $\|\mathbf{A}\|$ and $s_1(\mathbf{A})$, respectively.

For positive integers n, q, let $[n]$ denote the set $\{1, ..., n\}$ and $\mathbb{Z}_q$ denote the ring of integers modulo q. For a distribution or a set X, we write $x \xleftarrow{\$} X$ to denote the operation of sampling an uniformly random x according to X. For two distributions X, Y, we let $\mathsf{SD}(X, Y)$ denote their statistical distance. We write $X \overset{s}{\approx} Y$ to mean that they are statistically close, and $X \overset{c}{\approx} Y$ to say that they are computationally indistinguishable.

2.1 Lattice Trapdoor and Gaussian Sampling

The gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ is a primitive matrix defined by gadget vector $\mathbf{g}$ as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^\top \in \mathbb{Z}_q^{n\times nk}$. We usually consider gadget vector $\mathbf{g}^\top := [1\ 2\ 4\ \cdots\ 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$, where $k = \lceil \log_2 q \rceil$.

Gaussian Samplings.

Lemma 2.1 (TrapGen [MP12]) *Let q, n, m be positive integers with $q \geq 2$ and $m = O(n \log q)$. There is a PPT algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ that with overwhelming probability (in n) outputs a pair $(\mathbf{A}, \mathbf{T})$ such that $\mathbf{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and $\mathbf{T}$ is a basis for $\Lambda^\perp(\mathbf{A})$ satisfying*

$$\left\|\widetilde{\mathbf{T}}\right\| \leq O(\sqrt{n \log q}) \text{ and } \|\mathbf{T}\| \leq O(n \log q).$$

Lemma 2.2 (SampleLeft [ABB10]) *Let $q > 2$, $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ be two full rank matrices with $m > n$, $\mathbf{T_A}$ be a trapdoor matrix for $\mathbf{A}$, a matrix $\mathbf{U} \in \mathbb{Z}_q^{n\times l}$ and $s \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$. Then there exists a PPT algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \mathbf{U}, s)$ that outputs a matrix $\mathbf{K} \in \mathbb{Z}_q^{2m\times l}$, which is distributed statistically close to $\mathcal{D}_{\Lambda^{\mathbf{U}}\mathbb{Z}(\mathbf{A}|\mathbf{B}),s}$.*

Lemma 2.3 (SampleRight [MP12]) *Let $q > 2$, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be a full rank matrices with $m > n$, $\mathbf{R} \in \mathbb{Z}^{m\times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n\times l}$, $\gamma \in \mathbb{Z}_q$ with $\gamma \neq 0$ and $s \geq \sqrt{5} \cdot s_1(\mathbf{R}) \cdot \omega\sqrt{\log m}$. Then there exists a PPT algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}, \mathbf{U}, s)$ that outputs a matrix $\mathbf{K} \in \mathbb{Z}_q^{2m\times l}$, which is distributed statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{U}}(\mathbf{A}|\mathbf{A}\cdot\mathbf{R}+\gamma\mathbf{G}),s}$.*

Lemma 2.4 ([GPV08]) *For any prime q, integers $n \geq 1$, $m \geq 2n \log q$, $s \geq \omega(\sqrt{\log m})$, the following two distributions are statistically indistinguishable:*

- $(\mathbf{A}, \mathbf{u}, \mathbf{y})$*:* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{y} \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{u}},s}$.
- $(\mathbf{A}, \mathbf{u}, \mathbf{y})$*:* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$, $\mathbf{u} = \mathbf{A}\mathbf{y} \mod q$.

Lemma 2.5 (Noise Rerandomization [KY16]) *Let q, ℓ, m be positive integers and r a positive real satisfying $r > \max\{\eta_\epsilon(\mathbb{Z}^m), \eta_\epsilon(\mathbb{Z}^\ell)\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{x}$ chosen from $\mathcal{D}_{\mathbb{Z}^m,r}$. Then for any $\mathbf{V} \in \mathbb{Z}^{m\times \ell}$ and positive real $\sigma > s_1(\mathbf{V})$, there exists a PPT algorithm $\mathsf{ReRand}(\mathbf{V}, \mathbf{b}+\mathbf{x}, r, \sigma)$ that outputs $\mathbf{b}' = \mathbf{b}\mathbf{V} + \mathbf{x}'$ where the statistical distance of the discrete Gaussian $\mathcal{D}_{\mathbb{Z}^\ell,2r\sigma}$ and the distribution of $\mathbf{x}'$ is within 8ϵ.*

Theorem 2.1 (Two-Stage Sampling Algorithm [LLW21]) *For integers $q \geq 2$, $n \geq 1$, sufficiently large $m = O(n \log q)$, any $\mathbf{R} \in \mathbb{Z}^{m\times m}$, $s \geq \omega\sqrt{\log m}$ and $\rho \geq s\sqrt{m}\|\mathbf{R}\| \cdot \lambda^{\omega(1)}$, the output distributions $(\mathbf{A}, \mathbf{A}\mathbf{R}, \mathbf{y}, \mathbf{u})$ of the following two procedures are statistically close.*
$\mathsf{Sampler\text{-}1}$ $(\mathbf{R}, \rho, s)$*: Given a matrix $\mathbf{R} \in \mathbb{Z}^{m\times m}$ and two values $\rho, s \in \mathbb{R}$ as input, this sampler conducts the following steps in two stages.*

1. *Stage 1: (without the need of* $\mathbf{R}$*)*
   - *Sample a random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ *and its trapdoor* $\mathbf{T_A}$ *using* $\mathsf{TrapGen}(1^n, 1^m, q)$*;*
   - *Sample a random vector* $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$*;*
2. *Stage 2:*
   - *Sample a random vector* $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\rho}$*;*
   - *Sample a vector* $\mathbf{z}' = \begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{AR}, \mathbf{T_A}, \mathbf{u}-\mathbf{Ax}, s)$*, such that* $(\mathbf{A}|\mathbf{AR})\begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} = \mathbf{u}-\mathbf{Ax} \mod q$*;*
   - *Set* $\mathbf{y} = \begin{pmatrix}\mathbf{x}+\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} \in \mathbb{Z}^{2m}$*, satisfying* $(\mathbf{A}|\mathbf{AR})\mathbf{y} = \mathbf{u} \mod q$*;*
   - *Output the tuple* $(\mathbf{A}, \mathbf{AR}, \mathbf{y}, \mathbf{u})$.

$\mathsf{Sampler}$-2 $(\mathbf{R}, \rho, s)$*: Given a matrix* $\mathbf{R} \in \mathbb{Z}^{m\times m}$ *and two values* $\rho, s \in \mathbb{R}$ *as input, this sampler conducts the following steps in two stages.*

1. *Stage 1: (without the need of* $\mathbf{R}$*)*
   - *Sample a random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$*;*
   - *Sample a random vector* $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sqrt{\rho^2+s^2}}$ *and set* $\mathbf{u} = \mathbf{Ax} \mod q$*;*
2. *Stage 2:*
   - *Sample a random vector* $\mathbf{z}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$*;*
   - *Compute* $\mathbf{y} = \begin{pmatrix}\mathbf{x}-\mathbf{R}\mathbf{z}_2\\ \mathbf{z}_2\end{pmatrix} \in \mathbb{Z}^{2m}$*, satisfying* $(\mathbf{A}|\mathbf{AR})\mathbf{y} = \mathbf{u} \mod q$*;*
   - *Output the tuple* $(\mathbf{A}, \mathbf{AR}, \mathbf{y}, \mathbf{u})$.

We also provide a variant of two-stage sampling algorithm to support Q-tuples of output distributions which is useful in proving security for bounded collusion setting, named by multi-output two-stage sampling algorithm. The details of the algorithm and its proof can be found in Appendix A.2.

We will also need the following lemmas.

Lemma 2.6 (Leftover Hash Lemma [ABB10]) *Suppose that* $m > (n+1)\log q + \omega(\log n)$ *and that* $q > 2$ *is prime. Let* $\mathbf{R}$ *be an* $m\times k$ *matrix chosen uniformly in* $\{-1,1\}^{m\times k} \mod q$*, where* $k = k(n)$ *is polynomial in* n*. Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be matrices chosen uniformly in* $\mathbb{Z}_q^{n\times m}$ *and* $\mathbb{Z}_q^{n\times k}$ *respectively. Then, for all vectors* $\mathbf{e} \in \mathbb{Z}_q^m$*, the distribution* $(\mathbf{A}, \mathbf{AR}, \mathbf{R}^\top\mathbf{e})$ *is statistically close to the distribution* $(\mathbf{A}, \mathbf{B}, \mathbf{R}^\top\mathbf{e})$.

Lemma 2.7 (Cover-free Set Sampling Algorithm [LLW21]) *Let* $N = Qw\kappa^2$ *and* $w = \Theta(\kappa)$*. There exists an efficient sampler* $\mathsf{SamplerSet}(N, Q, v)$ *with the following properties: (1) The sampler always outputs a set* $\Delta \subset [N]$ *with cardinality* w*; (2) For independent samples* $\Delta_1, \ldots, \Delta_Q$ *from* $\mathsf{SamplerSet}(N, Q, w)$*, the sets are cover-free with probability* $1 - 2^{-\Omega(\kappa)}$*, i.e., for all* $i \in [Q]$, $\Pr[\Delta_i \backslash (\cup_{j\neq i}\Delta_j) \neq \emptyset] \geq 1 - 2Q \cdot 2^{-\Omega(\kappa)}$.

2.2 Leveled Fully Homomorphic Encryption

We now review the key and ciphertext formats of the leveled FHE scheme from [GSW13], along with the specific properties relevant to our constructions.

Lemma 2.8 (Leveled FHE [GSW13]) *In the leveled fully homomorphic encryption scheme in [GSW13], the keys and ciphertexts are formed as follows:*

- *The public key is* $\mathbf{A} = \begin{pmatrix}\mathbf{B}\\ \mathbf{r}^\top\mathbf{B}+\mathbf{e}^\top\end{pmatrix}$*, where* $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$*,* $\mathbf{r} \in \mathbb{Z}_q^n$ *and* $\mathbf{e} \in \mathbb{Z}^m$*. The secret key is* $\mathbf{s}^\top = (\mathbf{r}^\top, -1) \in \mathbb{Z}^{n+1}$.

- *A ciphertext of $x \in \{0,1\}$ is*

$$\Psi = \mathbf{A}\mathbf{R} + x\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m},$$

where $\mathbf{R}$ is the encryption randomness.

The decryption procedure for a ciphertext Ψ relies on the equation that

$$\mathbf{s}^\top \Psi = -\mathbf{e}^\top \mathbf{R} + x \cdot \mathbf{s}^\top \mathbf{G}.$$

The plaintext x can be further extracted via multiplication by $\mathbf{G}^{-1}(\lfloor q/2 \rceil \mathbf{e}_{n+1})$, where $\mathbf{e}_{n+1}$ is $(n+1)$-th canonical vector.

- *Suppose $\Psi_i = \mathbf{A}\mathbf{R}_i + x_i\mathbf{G}$ for $i \in [\ell]$ with $\mathbf{x} \in \{0,1\}^\ell$, then for a Boolean circuit $C : \{0,1\}^\ell \to \{0,1\}$, the ciphertext Ψ_C generated by HEval_C is*

$$\Psi_C = \mathbf{A}\mathbf{R}_C + C(\mathbf{x})\mathbf{G},$$

where $\|\mathbf{R}_C\|_\infty \leq (n \log q)^{O(d_C)} \max_{i\in[\ell]} \|\mathbf{R}_i\|$. The depth of HEval_C is $d_C \cdot O(\log m \log\log q)$.

2.3 Lattice Evaluation Algorithms

We use the evaluation algorithms for Boolean circuits as proposed in [BGG+14] and later extended to matrix-valued circuits in [BTVW17].

Theorem 2.2 (Attribute Encoding and Homomorphic Evaluations) *The attribute encoding and its homomorphic evaluation work as follows:*

- *The attribute encoding matrix used to encode attributes $\mathbf{x} \in 0,1^\ell$ is denoted as $\mathbf{A}_{\mathsf{attr}} \in \mathbb{Z}_q^{n\times \ell m}$.*
- *There exist efficient deterministic algorithms EvalF and EvalFX [BGG+14] such that for all $n, q, \ell \in \mathbb{N}$, any depth-$d$ Boolean circuit $f : \{0,1\}^\ell \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^\ell$, it holds that:*

$$\begin{aligned}
&\mathsf{EvalF}(\mathbf{A}_{\mathsf{attr}}, f) = \mathbf{H}_f \in \mathbb{Z}^{\ell m\times m}\\
&\mathsf{EvalFX}(\mathbf{A}_{\mathsf{attr}}, f, \mathbf{x}) = \mathbf{H}_{f,\mathbf{x}} \in \mathbb{Z}^{\ell m\times m}\\
&[\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}] \cdot \mathbf{H}_{f,\mathbf{x}} = \mathbf{A}_{\mathsf{attr}}\mathbf{H}_f + f(\mathbf{x})\mathbf{G}
\end{aligned}$$

Specifically, the norm of $\mathbf{H}_f$ and $\mathbf{H}_{f,\mathbf{x}}$ is bounded by $(n \log q)^{O(d)}$.

- *There exist efficient deterministic algorithms MEvalF and $\mathsf{MEvalFX}$ [BTVW17] such that for all $n, q, \ell \in \mathbb{N}$, any depth-$d$ matrix-valued circuit $f : \{0,1\}^\ell \mapsto \mathbf{X}_f \in \mathbb{Z}^{n\times m}$ and $\mathbf{x} \in \{0,1\}^\ell$, it holds that:*

$$\begin{aligned}
&\mathsf{MEvalF}(\mathbf{A}_{\mathsf{attr}}, f) = \mathbf{H}_f \in \mathbb{Z}^{\ell m\times m}\\
&\mathsf{MEvalFX}(\mathbf{A}_{\mathsf{attr}}, f, \mathbf{x}) = \mathbf{H}_{f,\mathbf{x}} \in \mathbb{Z}^{\ell m\times m}\\
&[\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}] \cdot \mathbf{H}_{f,\mathbf{x}} = \mathbf{A}_{\mathsf{attr}}\mathbf{H}_f + \mathbf{X}_f
\end{aligned}$$

Specifically, the norm of $\mathbf{H}_f$ and $\mathbf{H}_{f,\mathbf{x}}$ is bounded by $(n \log q)^{O(d)}\lceil \log q \rceil$.

Dual-Use Technique [BTVW17]. In essence, the dual-use of the secret $\mathbf{s}$ in both generating FHE ciphertexts and attribute encodings enables automatic decryption during homomorphic evaluations.

In more detail, let $f : \{0,1\}^\ell \to \{0,1\}$ be a Boolean circuit, $\mathbf{x} \in \{0,1\}^\ell$ be a bit-string. Each bit x_i of $\mathbf{x}$ is encrypted using leveled FHE encryption [GSW13] as Ψ_i. Given an attribute-encoding matrix $\mathbf{A}_{\mathsf{attr}} \in \mathbb{Z}_q^{n\times Lm}$ and $\psi \in \{0,1\}^L$ being as the bit-representation of $\Psi = (\Psi_1, \ldots, \Psi_\ell)$, the encoding of ψ is computed as follows:

$$\mathbf{s}^\top[\mathbf{A}_{\mathsf{attr}} + \psi^\top \otimes \mathbf{G}] + \mathbf{e}_{\mathsf{attr}}^\top ,$$

where the secret $\mathbf{s}$ is also served as the secret for encrypting Ψ_i.

Compute the matrix $\mathbf{H}_{\hat{\mathsf{HEval}}_f,\psi}$ for circuit $\hat{\mathsf{HEval}}_f$ that takes input as (the bit-representation of) Ψ and outputs $\overline{\Psi}_f$, where $\overline{\Psi}_f$ denotes all but last row of Ψ_f, using $\mathsf{MEvalFX}$ as follows:

$$\mathbf{H}_{\hat{\mathsf{HEval}}_f,\psi} := \mathsf{MEvalFX}(\mathbf{A}_{\mathsf{attr}}, \hat{\mathsf{HEval}}_f, \psi)$$

Thus, it holds that

$$\begin{aligned}
&(\mathbf{s}^\top[\mathbf{A}_{\mathsf{attr}} + \psi^\top \otimes \mathbf{G}] + \mathbf{e}_{\mathsf{attr}}^\top) \cdot \mathbf{H}_{\hat{\mathsf{HEval}}_f,\psi} - \underline{\Psi}_f \\
&\downarrow \text{(by encoding homomorphism)} \\
\approx\ & \mathbf{s}^\top \mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_{\hat{\mathsf{HEval}}_f} + \mathbf{s}^\top \hat{\mathsf{HEval}}_f(\Psi) - \underline{\Psi}_f \\
&\downarrow \text{(by definition of } \hat{\mathsf{HEval}}_f) \\
\approx\ & \mathbf{s}^\top \mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_{\hat{\mathsf{HEval}}_f} + \underbrace{\mathbf{s}^\top \overline{\Psi}_f - \underline{\Psi}_f}_{\mathsf{HE.Dec}} \\
\approx\ & \mathbf{s}^\top (\mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_{\hat{\mathsf{HEval}}_f} + f(\mathbf{x})\mathbf{G})
\end{aligned}$$

2.4 Fine-grained Functional Encryption

We note that both the predicate encryption (PE) and predicate inner-product functional encryption (P-IPFE) can be captured within the framework of Fine-grained Functional Encryption. Concretely, we consider a special class of FE with a function class $\mathcal{F} = \mathcal{P} \times \mathcal{G}$ and an input space $\mathcal{U} = \mathcal{X} \times \mathcal{M}$, where $\mathcal{P}$, $\mathcal{G}$, $\mathcal{X}$, and $\mathcal{M}$ represent the predicate space, key function space, attribute space, and message space, respectively. The overall function operates as follows:

$$f_{P,g}(x, m) := \begin{cases} g(m) & \text{if } P(x) = \text{true} \\ \perp & \text{otherwise.} \end{cases}$$

To capture PE, we define the key function as the identity function, i.e., $g(m) = m$. For P-IPFE, both the key function and message are represented by vectors. Specifically, for $\mathbf{v} \in \mathcal{G}$ and $\mathbf{w} \in \mathcal{M}$, the overall function is defined as follows:

$$f_{P,\mathbf{v}}(x, \mathbf{w}) := \begin{cases} \langle \mathbf{v}, \mathbf{w} \rangle & \text{if } P(x) = \text{true} \\ \perp & \text{otherwise.} \end{cases}$$

Next, we describe the formal definition for FE with fine-grained syntax.

Definition 1 (Fine-grained Functional Encryption) *A Functional Encryption scheme* FE *with fine-grained syntax for a family* $\mathcal{F}$*, defined by a predicate space* $\mathcal{P}$*, key function space* $\mathcal{G}$*, attribute space* $\mathcal{X}$ *and message space* $\mathcal{M}$*, consists of four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Setup}(1^\lambda, \mathcal{F}) \to (\mathsf{mpk}, \mathsf{msk})$*: On input the security parameter* λ *and a description of the function family* $\mathcal{F}$*, it outputs master public and secret keys* $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, P, g) \to \mathsf{sk}_{P,g}$*: On input the master secret key* msk*, a predicate* $P \in \mathcal{P}$ *and a key function* $g \in \mathcal{G}$*, it outputs a secret key* $\mathsf{sk}_{P,g}$.
- $\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}$*: On input the master public key* mpk*, an attribute* $x \in \mathcal{X}$ *and a message* $m \in \mathcal{M}$*, it outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}_{P,g}, \mathsf{ct}) \to \mu/\perp$*: On input a secret key* $\mathsf{sk}_{P,g}$ *a ciphertext* ct*, it outputs either a function value* μ *or* $\perp$.

Correctness. For all $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, any pair of attribute-message $(x, m) \in \mathcal{X} \times \mathcal{M}$ and any pair of predicate-key function $(P, g) \in \mathcal{P} \times \mathcal{G}$, we require that

- For 1-keys, namely $P(x) = \text{true}$,

$$\Pr\left[\mu = g(m) \;\middle|\; \begin{array}{l} \mathsf{sk}_{P,g} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, P, g) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m) \\ \mu \leftarrow \mathsf{Dec}(\mathsf{sk}_{P,g}, \mathsf{ct}) \end{array}\right] = 1 - \mathsf{negl}(\lambda) \;\;,$$

- For 0-keys, namely $P(x) = \text{false}$,

$$\Pr\left[\mu = \perp \;\middle|\; \begin{array}{l} \mathsf{sk}_{P,g} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, P, g) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m) \\ \mu \leftarrow \mathsf{Dec}(\mathsf{sk}_{P,g}, \mathsf{ct}) \end{array}\right] = 1 - \mathsf{negl}(\lambda) \;\;,$$

where the probabilities are taken over the coins of the setup algorithm Setup, secret keys $\mathsf{sk}_{P,g} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, P, g)$ and ciphertexts $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m)$.

Security. We provide the security definition for the fine-grained FE scheme as follows. Specifically, the collusion bound in this work refers to the number of admissible 1-key queries, i.e., q_1 in the following definition.

Definition 2 ((q_1, q_0)-xx-SIM security) *Let* FE *be a functional encryption scheme for predicate-key function* $\mathcal{F} = \mathcal{P} \times \mathcal{G}$ *and attribute-message space* $\mathcal{U} = \mathcal{X} \times \mathcal{M}$. *For every stateful* PPT *adversary* Adv, *a stateful simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{KeyGen}^*_{\mathsf{pre}}, \mathsf{Enc}^*, \mathsf{KeyGen}^*_{\mathsf{post}})$ *and every* $xx \in \{\mathsf{sel}, \mathsf{sa}, \mathsf{ada}\}$, *consider the following two experiments described in Fig. 1.*

$\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{FE},\mathsf{Adv}}(1^\lambda)$:	$\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{FE},\mathsf{Sim}}(1^\lambda)$:
1: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$	1: $(\mathsf{mpk}, \mathsf{st}) \leftarrow \mathsf{Setup}^*(1^\lambda, \mathcal{F})$
2: $(x, m) \leftarrow \mathsf{Adv}^{\mathcal{O}\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$	2: $(x, m) \leftarrow \mathsf{Adv}^{\mathsf{KeyGen}^*_{\mathsf{pre}}(\mathsf{st},\cdot)}(\mathsf{mpk})$
	$\mathsf{st} := \mathsf{st} \cup \{(f_i, \mathsf{sk}_{f_i}, f_i(x, m)\}$
3: $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m)$	3: $\mathsf{ct}^* \leftarrow \mathsf{Enc}^*(\mathsf{st})$
4: $\alpha \leftarrow \mathsf{Adv}^{\mathcal{O}\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}^*)$	4: $\alpha \leftarrow \mathsf{Adv}^{\mathsf{KeyGen}^*_{\mathsf{post}}(\mathsf{st},\cdot)}(\mathsf{ct}^*)$

Fig. 1. Security experiments $\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{FE},\mathsf{Adv}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{FE},\mathsf{Sim}}(1^\lambda)$ for Definition 2

We present the following supplementary notes.

- *An adversary* Adv *is admissible if it queries at most* q_1 *1-keys and* q_0 *0-keys with respect to the challenge index* x *during the experiment.*
- $\{\mathsf{sel}, \mathsf{sa}, \mathsf{ada}\}$ *refers to selective, semi-adaptive and adaptive, respectively. Selective security is defined by requiring the adversary to announce the challenge attribute* x *before receiving the public key, whereas semi-selective security requires the adversary to send the challenge attribute* x *after receiving the public key but before submitting any key queries.*

The fine-grained functional encryption scheme FE *is said to be* (q_1, q_0)*-xx-SIM secure if there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{KeyGen}^*_{\mathsf{pre}}, \mathsf{Enc}^*, \mathsf{KeyGen}^*_{\mathsf{post}})$ *such that for every admissible PPT adversary* Adv*, the following two distributions are computationally indistinguishable.*

$$\left\{\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{FE},\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{FE},\mathsf{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

3 Constructions of Predicate Encryption

3.1 (1, poly) Predicate Encryption

The proposed scheme $(1, \mathsf{poly})$ PE is essentially built upon the $(0, \mathsf{poly})$ predicate encryption of [BTVW17], which achieves weak attribute-hiding security. However, the key generation process incorporates the two-stage sampling algorithm from [LLW21]. The dual-use technique of [BTVW17] enables the automatic decryption of encrypted attributes following homomorphic evaluation, thereby circumventing the "lazy-OR" operations used in [GVW15, Agr17, LLW21] and corresponding attacks on PE when allowing 1-keys. The novel integration of the dual-use technique and two-stage sampling techniques results in a predicate encryption scheme that not only ensures strong attribute-hiding directly but also achieves more compact efficiency compared to prior constructions. Finally, we prove the strong attribute-hiding security of the resulting scheme using a different approach than that employed in [BTVW17].

Notation. We use gadget matrices $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\log q}$ and write $\overline{\mathbf{G}} \in \mathbb{Z}_q^{n\times(n+1)\log q}$ to denote all but the last row of $\mathbf{G}$. Similarly, we denote the last row of FHE ciphertext Ψ by $\underline{\Psi}$ and all but the last row of that by $\overline{\Psi}$. Specifically, throughout the whole paper, we will work on a predicate function class $\mathcal{F} : \{0,1\}^\ell \to \{0,1\}$ of depth denoted by d.

We describe the construction below.

Construction 1 **($(1, \mathsf{poly})$ Predicate Encryption).**

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d)$ *Given as input the security parameter* λ*, the attribute length* ℓ*, and the depth of the circuit family* d*, does the following:*

1. *Choose public parameters* (q, ρ, s, s_B, s_D) *as described in the following parameter setting paragraph.*
2. *Sample* $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
3. *Choose random matrices*
$$\mathbf{B}_j \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times(n+1)\log q} \text{ for } j \in [L],\ \mathbf{P} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m},$$
where $L = \ell\cdot(n+1)^2\log^2 q$*,* $m = (n+1)\log q$.
4. *Output the public and master secret keys.*
$$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P}),\ \mathsf{msk} := \mathbf{T_B}$$

$\mathsf{KeyGen}(\mathsf{msk}, f)$ *Given as input the master secret key* msk *and a circuit* f*, does the following:*

1. *Let* $\hat{f}$ *denote the circuit computing* $\Psi \mapsto \overline{\Psi}_f$*, compute*
$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}),\ \mathbf{B}_{\hat{f}} := [\mathbf{B}_1|\ldots|\mathbf{B}_L]\cdot\mathbf{H}_{\hat{f}}.$$
2. *Sample* $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$.
3. *Sample* $\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{B}_{\hat{f}}, \mathbf{T_B}, \mathbf{P} - \mathbf{BJ}, s)$*, s.t.* $[\mathbf{B}|\mathbf{B}_{\hat{f}}]\cdot\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{P} - \mathbf{BJ} \mod q$.
4. *Let* $\mathbf{K}_f := \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} + \begin{bmatrix}\mathbf{J}\\ \mathbf{0}\end{bmatrix}$*, s.t.* $[\mathbf{B}|\mathbf{B}_{\hat{f}}]\cdot\mathbf{K}_f = \mathbf{P} \mod q$.
5. *Output* $\mathsf{sk}_f := \mathbf{K}_f$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ *Given as input the master public key, an attribute* $\mathbf{x} \in \{0,1\}^\ell$ *and a message* $\mu \in \{0,1\}$, *does the following:*

1. *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$ *and* $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$.
2. *Sample* $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m \times m}$ *for* $i \in [\ell]$ *and compute*
$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\top \mathbf{B} + \mathbf{e}^\top \end{pmatrix} \mathbf{R}_i + x_i \mathbf{G}$$
Let $\psi = (\psi_1, \ldots, \psi_L)$ *denote the bit-representation of* $\Psi := [\Psi_1 | \cdots | \Psi_\ell]$.
3. *Let* $\mathbf{b} = [0, \ldots, 0, \lceil q/2 \rceil \mu]^\top \in \mathbb{Z}_q^m$. *Compute*
$$\beta_0 := \mathbf{B}^\top \mathbf{s} + \mathbf{e},\ \kappa := \mathbf{P}^\top \mathbf{s} + \mathbf{e}' + \mathbf{b}.$$
4. *Sample* $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m \times m}$ *for* $j \in [L]$ *and compute*
$$\mathbf{c}_j := [\mathbf{B}_j + \psi_j \overline{\mathbf{G}}]^\top \mathbf{s} + \mathbf{W}_j^\top \mathbf{e}.$$
5. *Output the ciphertext* $\mathsf{ct} := (\Psi, \beta_0, \kappa, \{\mathbf{c}_j\}_{j \in [L]})$.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$ *Given as input a secret key and a ciphertext, does the following:*

1. *Let* $\hat{f}$ *denote the circuit computing* $\Psi \mapsto \overline{\Psi}_f$, *compute*
$$\Psi_f \leftarrow \mathsf{HEval}_f(\Psi),$$
$$\mathbf{H}_{\hat{f},\psi} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j \in [L]}, \hat{f}, \psi),$$
$$\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top | \ldots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f.$$
2. *Compute* $\eta = \kappa - \mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix}$.
3. *Round each coordinate of* η. *If* $[\mathsf{Round}(\eta[1]), \ldots, \mathsf{Round}(\eta[m])] = \mathbf{0}$ *then set* $\mu = \mathsf{Round}(\eta[m])$ *and output* μ. *Otherwise, output* $\perp$.

Correctness. According to the key relation,

$$[\mathbf{B}_1 + \psi_1 \overline{\mathbf{G}} | \ldots | \mathbf{B}_L + \psi_L \overline{\mathbf{G}}] \cdot \mathbf{H}_{\hat{f},\psi} = [\mathbf{B}_1 | \ldots | \mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}} + \overline{\Psi}_f = \mathbf{B}_{\hat{f}} + \overline{\Psi}_f.$$

Thus, we have

$$\begin{aligned}
\mathbf{c}_{\hat{f}}^\top &:= [\mathbf{c}_1^\top | \ldots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f \\
&= \mathbf{s}^\top [\mathbf{B}_1 + \psi_1 \overline{\mathbf{G}} | \ldots | \mathbf{B}_L + \psi_L \overline{\mathbf{G}}] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f + \mathbf{e}_{\mathsf{attr.Eval}} \\
&= \mathbf{s}^\top (\mathbf{B}_{\hat{f}} + \overline{\Psi}_f) - \underline{\Psi}_f + \mathbf{e}_{\mathsf{attr.Eval}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + [\mathbf{s}^\top | -1] \cdot \Psi_f + \mathbf{e}_{\mathsf{attr.Eval}}
\end{aligned}$$

The FHE ciphertext Ψ_f after homomorphic evaluation can be written as

$$\Psi_f = \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\top \mathbf{B} + \mathbf{e}^\top \end{pmatrix} \mathbf{R}_f + f(\mathbf{x}) \mathbf{G}.$$

Then, we have

$$\begin{aligned}
\mathbf{c}_{\hat{f}}^\top &= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + [\mathbf{s}^\top | -1] \cdot \Psi_f + \mathbf{e}_{\mathsf{attr.Eval}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + f(\mathbf{x}) \cdot [\mathbf{s}^\top | -1] \cdot \mathbf{G} + \underbrace{\mathbf{e}_{\mathsf{attr.Eval}} + \mathbf{e}_{\mathsf{HE.Eval}}}_{\mathbf{e}_{\mathsf{Eval}}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + \mathbf{e}_{\mathsf{Eval}} \text{ (when } f(\mathbf{x}) = 0)
\end{aligned}$$

Hence,

$$\mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} = \mathbf{P}^\top \mathbf{s} + \mathbf{K}_f^\top \begin{pmatrix} \mathbf{e} \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix}$$
$$\kappa - \mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} = \mathbf{b} + \underbrace{\left\{ \mathbf{e}' - \mathbf{K}_f^\top \begin{pmatrix} \mathbf{e} \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix} \right\}}_{\mathbf{e}_{\mathsf{dec}}}.$$

Thus, we require that when $f(\mathbf{x}) = 0$, the first $m-1$ coordinates of $\mathbf{e}_{\mathsf{dec}}$ to be bounded by $q/4$, which can be ensured by our parameter setting.

Parameters Setting. The parameters setting and the detailed proof of Theorem 3.1 can be found in Appendix B.1.

Security.

Theorem 3.1 *Assuming the hardness of* LWE*, then the construction 1 is a* PE *for the class $\mathcal{F}$, achieving* $(1, \mathsf{poly})$-sel-*SIM security that allows at most single 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

3.2 More compact Bounded Collusion-Resistant PE

Construction 2 **(**(Q, poly) **Predicate Encryption).**

$\mathsf{QPE.Setup}(1^\lambda, 1^\ell, 1^d, 1^Q)$ *Given as input the security parameter λ, the attribute length ℓ, the depth of the circuit family d and the upper bound of 1-key queries Q, does the following:*

1. *Choose public parameters $(q, \rho, s_B, s_D, N, w)$ as described in the following parameter setting paragraph.*
2. *Sample* $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
3. *Choose random matrices*
$$\mathbf{B}_j \xleftarrow{\$} \mathbb{Z}_q^{n \times (n+1)\log q} \text{ for } j \in [L],\ \mathbf{P}_k \xleftarrow{\$} \mathbb{Z}_q^{n \times m} \text{ for } k \in [N],$$
where $L = \ell \cdot (n+1)^2 \log^2 q$, $m = (n+1)\log q$.
4. *Output the public and master secret keys.*
$$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j \in [L]}, \{\mathbf{P}_k\}_{k \in [N]}),\ \mathsf{msk} := \mathbf{T_B}$$

$\mathsf{QPE.KeyGen}(\mathsf{msk}, f)$ *Given as input the master secret key* msk *and a circuit f, does the following:*

1. *Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute*
$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j \in [L]}, \hat{f}),\ \mathbf{B}_{\hat{f}} := [\mathbf{B}_1 | \ldots | \mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}}.$$
2. *Sample a random subset $\Delta \subset [N]$ according to sampler* $\mathsf{SamplerSet}(N, Q, w)$ *with $|\Delta| = w$, and compute the sum of the subset* $\mathbf{P}_\Delta = \sum_{k \in \Delta} \mathbf{P}_k$.
3. *Sample* $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m}, \rho}$.
4. *Sample* $\begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{B}_{\hat{f}}, \mathbf{T_B}, \mathbf{P}_\Delta - \mathbf{BJ}, s)$, *s.t.* $[\mathbf{B} | \mathbf{B}_{\hat{f}}] \cdot \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} = \mathbf{P}_\Delta - \mathbf{BJ} \mod q$.
5. *Let* $\mathbf{K}_f := \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \\ \mathbf{0} \end{bmatrix}$, *s.t.* $[\mathbf{B} | \mathbf{B}_{\hat{f}}] \cdot \mathbf{K}_f = \mathbf{P}_\Delta \mod q$.
6. *Output* $\mathsf{sk}_f := (\Delta, \mathbf{K}_f)$.

$\mathsf{QPE.Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ *Given as input the master public key, an attribute $\mathbf{x} \in \{0,1\}^\ell$ and a message $\mu \in \{0,1\}$, does the following:*

1. *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$ *and* $\mathbf{e}'_k \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ *for* $k \in [N]$.

2. *Sample* $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ *for* $i \in [l]$ *and compute*

$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\top \mathbf{B} + \mathbf{e}^\top \end{pmatrix} \mathbf{R}_i + x_i \mathbf{G}$$

Let $\psi = (\psi_1, \ldots, \psi_L)$ *denote the bit-representation of* $\Psi := [\Psi_1|\cdots|\Psi_\ell]$.

3. *Let* $\mathbf{b}_k = [0, \ldots, 0, \frac{\lceil q/2 \rceil}{w}\mu]^\top \in \mathbb{Z}_q^m$ *for* $k \in [N]$. *Compute*

$$\beta_0 := \mathbf{B}^\top \mathbf{s} + \mathbf{e},\ \beta_{1,k} := \mathbf{P}_k^\top \mathbf{s} + \mathbf{e}'_k + \mathbf{b}_k.$$

4. *Sample* $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ *for* $j \in [L]$ *and compute*

$$\mathbf{c}_j := [\mathbf{B}_j + \psi_j \overline{\mathbf{G}}]^\top \mathbf{s} + \mathbf{W}_j^\top \mathbf{e}.$$

5. *Output the ciphertext* $\mathsf{ct} := (\Psi, \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{QPE.Dec}(\mathsf{sk}_f, \mathsf{ct})$ *Given as input a secret key and a ciphertext, does the following:*

1. *Let* $\hat{f}$ *denote the circuit computing* $\Psi \mapsto \overline{\Psi}_f$, *compute*

$$\Psi_f \leftarrow \mathsf{HEval}_f(\Psi),$$
$$\mathbf{H}_{\hat{f},\psi} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}, \psi),$$
$$\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top|\ldots|\mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f.$$

2. *Compute* $\eta = \sum_{k\in\Delta} \beta_{1,k} - \mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix}$.

3. *Round each coordinate of* η. *If* $[\mathsf{Round}(\eta[1]), \ldots, \mathsf{Round}(\eta[m])] = \mathbf{0}$ *then set* $\mu = \mathsf{Round}(\eta[m])$ *and output* μ. *Otherwise, output* $\perp$.

Correctness. Similar to the $(1, \mathsf{poly})$ PE scheme, the encrypted attribute encoding after the homomorphic evaluation would take the following form:

$$\begin{aligned}
\mathbf{c}_{\hat{f}}^\top &= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + [\mathbf{s}^\top| -1] \cdot \Psi_f + \mathbf{e}_{\mathsf{attr.Eval}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + f(\mathbf{x}) \cdot [\mathbf{s}^\top| -1] \cdot \mathbf{G} + \underbrace{\mathbf{e}_{\mathsf{attr.Eval}} + \mathbf{e}_{\mathsf{HE.Eval}}}_{\mathbf{e}_{\mathsf{Eval}}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + \mathbf{e}_{\mathsf{Eval}} \text{ (when } f(\mathbf{x}) = 0)
\end{aligned}$$

Hence,

$$\mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} = \mathbf{P}_\Delta^\top \cdot \mathbf{s} + \mathbf{K}_f^\top \cdot \begin{pmatrix} \mathbf{e} \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix}$$

$$\textstyle\sum_{k\in\Delta} \beta_{1,k} - \mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} = \mathbf{b} + \underbrace{\left\{ \sum_{k\in\Delta} \mathbf{e}'_k - \mathbf{K}_f^\top \begin{pmatrix} \mathbf{e} \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix} \right\}}_{\mathbf{e}_{\mathsf{dec}}}.$$

Thus, we require that when $f(\mathbf{x}) = 0$, the first $m - 1$ coordinates of $\mathbf{e}_{\mathsf{dec}}$ to be bounded by $q/4$, which can be ensured by our parameter setting.

Parameters Setting. The parameters setting and the detailed proof of Theorem 3.2 can be found in Appendix B.2.

Security.

Theorem 3.2 *Assuming the hardness of* LWE, *then the construction 2 is a* PE *for the class* $\mathcal{F}$, *achieving* (Q, poly)-sel-*SIM security that allows up to* Q *1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

4 Constructions of Predicate Inner Product Encryption scheme

In this section, we begin by constructing a P-IPFE scheme that permits only a single pre-challenge 1-key query. In this case, the basic version of ALS IPFE scheme [ALS16] is sufficient. Next, we leverage techniques involving the extension of dimensions and cover-free sets to present a bounded collusion predicate IPFE scheme that relies on the security of a modified N-ALS IPFE construction [WFL19].

4.1 (1, poly) Predicate Inner Product Functional Encryption

We now introduce our $(1, \mathsf{poly})$ P-IPFE construction and prove its fully attribute-hiding (selective) security. Specifically, we consider the message vector space $\mathcal{U} = \{1, \ldots, U-1\}^t$ and the key vector space $\mathcal{V} = \{1, \ldots, V-1\}^t$ for some integer U, P and dimension $t = \mathsf{poly}(\lambda)$. The inner products are evaluated over $\mathbb{Z}$ and belongs to $\{1, \ldots, Y-1\}$ with $Y = tUV$.

Construction 3 ($(1, \mathsf{poly})$ **Predicate IPFE).**

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d, 1^t)$ *Given as input the security parameter λ, the attribute length ℓ, the depth d of the circuit family and the length of message (key) vector t, does the following:*

1. *Choose public parameters (q, s, ρ, s_B, s_D) as described in the following parameter setting paragraph.*
2. *Sample $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.*
3. *Choose random matrices*
$$\mathbf{B}_j \xleftarrow{\$} \mathbb{Z}_q^{n\times(n+1)\log q} \text{ for } j \in [L],\ \mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{n\times t},$$
where $L = \ell \cdot (n+1)^2 \log^2 q$, $m = (n+1)\log q$.
4. *Output the public and master secret keys.*
$$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P}),\ \mathsf{msk} := \mathbf{T_B}$$

$\mathsf{KeyGen}(\mathsf{msk}, f, \mathbf{v})$ *Given as input the master secret key* msk*, a circuit f and a key vector $\mathbf{v} \in \mathcal{V}$, does the following:*

1. *Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute*
$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}),\ \mathbf{B}_{\hat{f}} := [\mathbf{B}_1|\ldots|\mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}}.$$
2. *Sample $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times t},\rho}$.*
3. *Sample $\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{B}_{\hat{f}}, \mathbf{T_B}, \mathbf{P} - \mathbf{BJ}, s)$, s.t. $[\mathbf{B}|\mathbf{B}_{\hat{f}}] \cdot \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{P} - \mathbf{BJ} \mod q$.*
4. *Let $\mathbf{K}_f := \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} + \begin{bmatrix}\mathbf{J}\\ \mathbf{0}\end{bmatrix}$, s.t. $[\mathbf{B}|\mathbf{B}_{\hat{f}}] \cdot \mathbf{K}_f = \mathbf{P} \mod q$.*
5. *Output $\mathsf{sk}_{f,\mathbf{v}} := \mathbf{K}_f \cdot \mathbf{v}$.*

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mathbf{u})$ *Given as input the master public key, an attribute $\mathbf{x} \in \{0,1\}^\ell$ and a message vector $\mathbf{u} \in \mathcal{U}$, does the following:*

1. *Sample $\mathbf{s}, \mathbf{s}' \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0, \mathbf{e}_0' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$ and $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^t, s_D}$.*
2. *Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ for $i \in [\ell]$ and compute*
$$\Psi_i := \begin{pmatrix}\mathbf{B}\\ \mathbf{s}^\top\mathbf{B} + \mathbf{e}_0^\top\end{pmatrix}\mathbf{R}_i + x_i\mathbf{G},\ \ \Psi_i' := \begin{pmatrix}\mathbf{B}\\ \mathbf{s}'^\top\mathbf{B} + \mathbf{e}_0'^\top\end{pmatrix}\mathbf{R}_i + x_i\mathbf{G},$$
Let $\psi' = (\psi_1', \ldots, \psi_L')$ denote the bit-representation of $\Psi' := [\Psi_1'|\cdots|\Psi_\ell']$.
3. *Compute*
$$\beta_0 := \mathbf{B}^\top\mathbf{s} + \mathbf{e}_0,\ \beta_1 := \mathbf{P}^\top\mathbf{s} + \mathbf{e}_1 + \left\lfloor \tfrac{q}{Y} \right\rfloor \cdot \mathbf{u}.$$

4. Sample $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ *for* $j \in [L]$ *and compute*

$$\mathbf{c}_j := [\mathbf{B}_j + \psi_j \overline{\mathbf{G}}]^\top \mathbf{s} + \mathbf{W}_j^\top \mathbf{e}_0.$$

5. Output the ciphertext $\mathsf{ct} := (\Psi, \Psi', \beta_0, \beta_1, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *Given as input a secret key and a ciphertext, does the following:*

1. Compute $\mathsf{HEval}_f(\Psi) = \Psi_f = \begin{pmatrix} \overline{\Psi}_f \\ \underline{\Psi}_f \end{pmatrix}$.

2. Let $\hat{f}$ *denote the circuit computing* $\Psi \mapsto \overline{\Psi}_f$*, compute*

$$\mathbf{H}_{\hat{f},\psi'} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}, \psi'),$$
$$\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top | \dots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f.$$

3. Compute $\mu' = \langle \beta_1, \mathbf{v}\rangle - \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix}^\top \cdot \mathsf{sk}_{f,\mathbf{v}} \mod q$.

4. Output the value $\mu \in \{-Y+1, \dots, Y-1\}$ *that minimizes* $|\lfloor \frac{q}{Y} \rfloor \cdot \mu - \mu'|$.

Correctness. Notice that the FHE ciphertexts Ψ and Ψ' share the same encryption randomness and public matrix $\mathbf{B}$, thus we have

$$\Psi_f = \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\top \mathbf{B} + \mathbf{e}_0^\top \end{pmatrix} \mathbf{R}_f + f(\mathbf{x})\mathbf{G},\ \Psi'_f = \begin{pmatrix} \mathbf{B} \\ \mathbf{s}'^\top \mathbf{B} + \mathbf{e}_0'^\top \end{pmatrix} \mathbf{R}_f + f(\mathbf{x})\mathbf{G}.$$

In other words, $\overline{\Psi}_f = \overline{\Psi}'_f$. Thus, the homomorphic evaluation results as follows:

$$\begin{aligned}
\mathbf{c}_{\hat{f}}^\top &:= [\mathbf{c}_1^\top | \dots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f \\
&= \mathbf{s}^\top (\mathbf{B}_{\hat{f}} + \overline{\Psi}'_f) - \underline{\Psi}_f + \mathbf{e}_{\mathsf{attr.Eval}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + [\mathbf{s}^\top | -1] \cdot \Psi_f + \mathbf{e}_{\mathsf{attr.Eval}} \\
&= \mathbf{s}^\top \mathbf{B}_{\hat{f}} + \underbrace{\mathbf{e}_{\mathsf{attr.Eval}} + \mathbf{e}_{\mathsf{HE.Eval}}}_{\mathbf{e}_{\mathsf{Eval}}} \text{ (when } f(\mathbf{x}) = 0)
\end{aligned}$$

Hence,

$$\begin{aligned}
&\langle \beta_1, \mathbf{v}\rangle \\
&= (\mathbf{P}^\top \mathbf{s} + \mathbf{e}_1 + \left\lfloor \frac{q}{Y} \right\rfloor \mathbf{u})^\top \cdot \mathbf{v} \\
&= \mathbf{s}^\top \mathbf{P} \cdot \mathbf{v} + \langle \mathbf{e}_1, \mathbf{v}\rangle + \left\lfloor \frac{q}{Y} \right\rfloor \langle \mathbf{u}, \mathbf{v}\rangle,
\end{aligned}$$

$$\begin{aligned}
&\begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix}^\top \cdot \mathsf{sk}_{f,\mathbf{v}} \\
&= (\mathbf{s}^\top \mathbf{B} + \mathbf{e}_0 \mid \mathbf{s}^\top \mathbf{B}_{\hat{f}} + \mathbf{e}_{\mathsf{Eval}}) \cdot \mathbf{K}_f \cdot \mathbf{v} \\
&= \mathbf{s}^\top \mathbf{P} \cdot \mathbf{v} + \begin{pmatrix} \mathbf{e}_0 \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix}^\top \mathbf{K}_f \cdot \mathbf{v}.
\end{aligned}$$

It is easy to check that when $f(\mathbf{x}) = 0$,

$$\begin{aligned}
\mu' &= \left\lfloor \frac{q}{Y} \right\rfloor \langle \mathbf{u}, \mathbf{v}\rangle + \underbrace{\langle \mathbf{e}_1, \mathbf{v}\rangle - \begin{pmatrix} \mathbf{e}_0 \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix}^\top \mathbf{K}_f \cdot \mathbf{v}}_{\mathbf{e}_{\mathsf{dec}}} \\
&= \left\lfloor \frac{q}{Y} \right\rfloor \langle \mathbf{u}, \mathbf{v}\rangle + e_{\mathsf{dec}}.
\end{aligned}$$

If the magnitude of error term e_{dec} is bounded by $q/2Y$ with overwhelming probability, which can be ensured by our parameter setting, then the correctness holds with overwhelming probability.

Parameters Setting. The parameters setting and the detailed proof of Theorem 4.1 can be found in Appendix C.1.

Security.

Theorem 4.1 *Assuming the hardness of* LWE*, then the scheme described in Section 4.1 is a* P-IPFE *for the predicate class $\mathcal{F}$, message vector space $\mathcal{U}$ and key vector space $\mathcal{V}$, achieving* $(1, \mathsf{poly})$-sel-*SIM security that allows at most single 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

4.2 (Q, poly)-Predicate Inner Product Functional Encryption

We now propose a predicate IPFE scheme that allows up to Q 1-key queries and any polynomial number of 0-keys as follows. Specifically, we consider the inner products modulo prime p, hence, the plaintext and key vectors belong to $\mathbb{Z}_p^t$.

Construction 4 **((Q, poly) Predicate IPFE).**

$\mathsf{QSetup}(1^\lambda, 1^\ell, 1^d, 1^t, 1^Q)$ *Given as input the security parameter λ, the attribute length ℓ, the depth of the circuit family d, the length of message (key) vector t and the upper bound of 1-key queries Q, does the following:*

1. *Choose public parameters $(q, \rho, s, s_B, s_D, N, w)$ as described in the following parameter setting paragraph.*
2. *Sample $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.*
3. *Choose random matrices*
$$\mathbf{B}_j \xleftarrow{\$} \mathbb{Z}_q^{n\times(n+1)\log q} \textit{ for } j \in [L],\ \mathbf{P}_k \xleftarrow{\$} \mathbb{Z}_q^{n\times(t+2)} \textit{ for } k \in [N],$$
where $L = \ell \cdot (n+1)^2 \log^2 q$, $m = (n+1)\log q$.
4. *Output the public and master secret keys.*
$$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \{\mathbf{P}_k\}_{k\in[N]}),\ \mathsf{msk} := \mathbf{T_B}$$

$\mathsf{QKeyGen}(\mathsf{msk}, f, \mathbf{v})$ *Given as input the master secret key* msk*, a circuit f and a key vector $\mathbf{v} \in \mathbb{Z}_p^t$, does the following:*

1. *Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute*
$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}),\ \mathbf{B}_{\hat{f}} := [\mathbf{B}_1|\dots|\mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}}.$$
2. *Sample a random subset $\Delta \subset [N]$ according to sampler $\mathsf{SamplerSet}(N, Q, w)$ with $|\Delta| = w$, and compute the sum of the subset $\mathbf{P}_\Delta = \sum_{k\in\Delta} \mathbf{P}_k$.*
3. *Sample $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times(t+2)},\rho}$.*
4. *Sample $\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{B}_{\hat{f}}, \mathbf{T_B}, \mathbf{P}_\Delta - \mathbf{BJ}, s)$, s.t. $[\mathbf{B}|\mathbf{B}_{\hat{f}}] \cdot \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{P}_\Delta - \mathbf{BJ} \mod q$.*
5. *Let $\mathbf{K}_f := \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} + \begin{bmatrix}\mathbf{J}\\ \mathbf{0}\end{bmatrix}$, s.t. $[\mathbf{B}|\mathbf{B}_{\hat{f}}] \cdot \mathbf{K}_f = \mathbf{P}_\Delta \mod q$.*
6. *Sample $r \xleftarrow{\$} \mathbb{Z}_p$, set $\mathbf{v}'^\top = (\mathbf{v}^\top, 1, r)^\top$.*
7. *Compute $\mathsf{sk}_{f,\mathbf{v}} := (\Delta, \mathbf{v}', \mathbf{k}_{f,\mathbf{v}} := \mathbf{K}_f \cdot \mathbf{v}')$.*

$\mathsf{QEnc}(\mathsf{mpk}, \mathbf{x}, \mathbf{u})$ *Given as input the master public key, an attribute $\mathbf{x} \in \{0,1\}^\ell$ and a message vector $\mathbf{u}$, does the following:*

1. *Sample $\mathbf{s}, \mathbf{s}' \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0, \mathbf{e}'_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$ and $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^{t+2}, s_D}$ for $k \in [N]$.*
2. *Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ for $i \in [\ell]$ and compute*

$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\top \mathbf{B} + \mathbf{e}_0^\top \end{pmatrix} \mathbf{R}_i + x_i \mathbf{G}, \;\; \Psi'_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{s}'^\top \mathbf{B} + \mathbf{e}_0'^\top \end{pmatrix} \mathbf{R}_i + x_i \mathbf{G},$$

Let $\psi' = (\psi'_1, \ldots, \psi'_L)$ *denote the bit-representation of* $\Psi' := [\Psi'_1 | \cdots | \Psi'_\ell]$.

3. *Set* $\mathbf{u}'_k := (\frac{1}{w}\mathbf{u}^\top, 0, 0)^\top \in \mathbb{Z}_p^{t+2}$ *for* $k \in [N]$. *Compute*
$$\beta_0 := \mathbf{B}^\top \mathbf{s} + \mathbf{e}_0, \; \beta_{1,k} := \mathbf{P}_k^\top \mathbf{s} + \mathbf{e}_{1,k} + p^{e-1}\mathbf{u}'_k.$$
4. *Sample* $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m \times m}$ *for* $j \in [L]$ *and compute*
$$\mathbf{c}_j := [\mathbf{B}_j + \psi'_j \overline{\mathbf{G}}]^\top \mathbf{s} + \mathbf{W}_j^\top \mathbf{e}_0.$$
5. *Output the ciphertext* $\mathsf{ct} := (\Psi, \Psi', \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{QDec}(\mathsf{sk}_{f,\mathbf{v}}, \mathsf{ct})$ *Given as input a secret key and a ciphertext, does the following:*

1. *Compute* $\mathsf{HEval}_f(\Psi) = \Psi_f = \begin{pmatrix} \overline{\Psi}_f \\ \underline{\Psi}_f \end{pmatrix}$.
2. *Let* $\hat{f}$ *denote the circuit computing* $\Psi \mapsto \overline{\Psi}_f$, *compute*
$$\mathbf{H}_{\hat{f},\psi'} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}, \psi'),$$
$$\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top | \ldots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f.$$
3. *Compute* $\mu' = \langle \sum_{k\in\Delta} \beta_{1,k}, \mathbf{v}' \rangle - \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix}^\top \cdot \mathbf{k}_{f,\mathbf{v}} \mod q$.
4. *Output the value* $\mu \in \mathbb{Z}_p$ *that minimizes* $|p^{e-1} \cdot \mu - \mu'|$.

Correctness. Essentially, the attribute encoding components in the ciphertext are the same as the $(1, \mathsf{poly})$ P-IPFE scheme in Section 4.1. Therefore, when $f(\mathbf{x}) = 0$, we could easily obtain $\mathbf{c}_{\hat{f}}^\top = \mathbf{s}^\top \mathbf{B}_{\hat{f}} + \mathbf{e}_{\mathsf{Eval}}$. Moreover, we have

$$\begin{aligned}
&\langle \sum_{k\in\Delta} \beta_{1,k}, \mathbf{v}' \rangle \\
&= (\sum_{k\in\Delta} \mathbf{P}_k^\top \mathbf{s} + \mathbf{e}_{1,k} + p^{e-1}\mathbf{u}'_k)^\top \cdot \mathbf{v}' \\
&= \mathbf{s}^\top \mathbf{P}_\Delta \cdot \mathbf{v}' + \underbrace{\left(\sum_{k\in\Delta} \mathbf{e}_{1,k}\right)^\top}_{\mathbf{e}_1} \cdot \mathbf{v}' + p^{e-1} \underbrace{\left(\sum_{k\in\Delta} \mathbf{u}'_k\right)^\top}_{(\mathbf{u}^\top, 0, 0)} \cdot \mathbf{v}',
\end{aligned}$$

along with

$$\begin{aligned}
&\begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix}^\top \cdot \mathbf{k}_{f,\mathbf{v}} \\
&= (\mathbf{s}^\top \mathbf{B} + \mathbf{e})_0 \mid \mathbf{s}^\top \mathbf{B}_{\hat{f}} + \mathbf{e}_{\mathsf{Eval}}) \cdot \mathbf{K}_f \cdot \mathbf{v}' \\
&= \mathbf{s}^\top \mathbf{P}_\Delta \cdot \mathbf{v}' + \begin{pmatrix} \mathbf{e}_0 \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix}^\top \mathbf{K}_f \cdot \mathbf{v}'.
\end{aligned}$$

Therefore, when $f(\mathbf{x}) = 0$, we have

$$\begin{aligned}
\mu' &= p^{e-1} \langle (\mathbf{u}^\top, 0, 0), (\mathbf{v}^\top, 1, r) \rangle + \underbrace{\langle \mathbf{e}_1, \mathbf{v}' \rangle - \begin{pmatrix} \mathbf{e}_0 \\ \mathbf{e}_{\mathsf{Eval}} \end{pmatrix}^\top \mathbf{K}_f \cdot \mathbf{v}'}_{\mathbf{e}_{\mathsf{dec}}} \\
&= p^{e-1} \langle \mathbf{u}, \mathbf{v} \rangle + \mathbf{e}_{\mathsf{dec}}.
\end{aligned}$$

If the magnitude of error term $\mathbf{e}_{\mathsf{dec}}$ is bounded by $p^{e-1}/2$ with overwhelming probability, which can be ensured by our parameter setting, then the correctness holds with overwhelming probability.

Parameters Setting. The parameters setting and the detailed proof of Theorem 4.2 can be found in Appendix C.2.

Security.

Theorem 4.2 *Assuming the hardness of* LWE*, then the scheme described in Section 4.2 is a* P-IPFE *for the predicate class $\mathcal{F}$, message vector space $\mathcal{U}$ and key vector space $\mathcal{V}$, achieving (Q, poly)-sel-SIM security that allows up to Q 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

For clarity of the presentation, we just describe the simulator Sim for Theorem 4.2 here, and defer the detailed proof to the full version.

Simulator. $\mathsf{QSim}^*(1^\lambda, 1^{|\mathbf{x}|}, 1^{|\mathbf{u}|})$:

1. $\mathsf{QSetup}^*(1^\lambda, 1^{|\mathbf{x}|}, 1^{|\mathbf{u}|})$: It generates all public parameters as in the real QSetup, except that it runs $(\mathbf{B}', \mathbf{T}_{\mathbf{B}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^m, q)$, then parse $\mathbf{B}' = \begin{bmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{bmatrix}$, where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, and sets $\mathbf{B}$ be the public matrix in mpk. Then, it initializes $\mathsf{st} := \emptyset$.
2. $\mathsf{QKeyGen}^*_{\mathsf{pre}}(\mathsf{st}, f, \mathbf{v})$: It generates all secret keys as in the real QKeyGen algorithm and simultaneously maintains st that contains $\{f_{\hat{i}}, \mathbf{v}_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}}, \mathbf{K}_{\hat{i}} \cdot \mathbf{v}'_{\hat{i}})\}_{\hat{i}\in[Q']}$ for $f_{\hat{i}}$ such that $f_{\hat{i}}(\mathbf{x}^*) = 0$.
3. $\mathsf{QEnc}^*(\mathsf{st})$: It takes as input st that contains $d^{\mathsf{pre}}_{\hat{i}} = \langle \mathbf{u}^*, \mathbf{v}^{\mathsf{pre}}_{\hat{i}} \rangle$ if the adversary has queried for $(f_{\hat{i}}, \mathbf{v}_{\hat{i}})$ such that $f_{\hat{i}}(\mathbf{x}^*) = 0$ before the challenge query, then constructs the challenge ciphertext as follows.
   (a) It samples $\beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ independently and uniformly from $\mathbb{Z}_q^m$.
   (b) Samples $\{\Psi_i, \Psi'_i\}_{i\in[\ell]}$ uniformly from $\mathbb{Z}_q^{(n+1)\times(n+1)\log q}$.
   (c) If $\mathsf{st} = \emptyset$, i.e., the adversary did not make any 1-key in the pre-challenge phase, it computes $\{\beta_{1,k}\}_{k\in[N]}$ as follows:
      - Choose Q random subset $(\Delta_1, \ldots, \Delta_Q)$ with size w according sampler $\mathsf{SamplerSet}(N, Q, w)$, sample $r_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_p$ for $\hat{i} \in [Q]$.
      - Generate random shares $\{r'_k\}_{k\in[N]}$ over $\mathbb{Z}_p$ under the following constraints: for $\hat{i} \in [Q]$, $\sum_{k\in\Delta_{\hat{i}}} r'_k = r_{\hat{i}}$. This can be done efficiently by the cover-freeness of the subsets, using the following standard procedure.
        Let $\delta_{\hat{i}}$ be a unique index that appears only in $\Delta_{\hat{i}}$ but not in the other subsets. To generate the random shares $\{r'_k\}_{k\in[N]}$, we first sample r'_k randomly for all $k \in [N]\backslash\{\delta_{\hat{i}}\}_{\hat{i}\in[Q]}$, and then fix $r'_{\delta_{\hat{i}}} = r_{\hat{i}} - \sum_{k\in\Delta_{\hat{i}}\backslash\{\delta_{\hat{i}}\}} r'_k$ for $\hat{i} \in [Q]$.
      - For $k \in [N]$, set $\mathbf{u}'_k = (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, 1)^\top \in \mathbb{Z}_p^{t+2}$ for $\widetilde{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^t$, sample $\widetilde{\beta}_k \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$.
      - Set $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}_{1,k} + \mathbf{u}'_k \mod q$.
   (d) Otherwise, if the adversary has submitted Q' 1-key queries in the pre-challenge phase, then update $\mathsf{st} = \mathsf{st} \parallel \{d^{\mathsf{pre}}_{\hat{i}} = \langle \mathbf{u}^*, \mathbf{v}^{\mathsf{pre}}_{\hat{i}} \rangle\}_{\hat{i}\in[Q']}$, then QEnc^* generates $\{\beta_{1,k}\}_{k\in[N]}$ to satisfy the decryption consistency as follows.
      - For $\hat{i} \in [Q']$, compute $\Psi_{f_{\hat{i}}} := \mathsf{HEval}_{f_{\hat{i}}}(\Psi)$. Let $\hat{f}_{\hat{i}}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_{f_{\hat{i}}}$, compute $\mathbf{H}_{\hat{f}_{\hat{i}},\Psi'} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}_{\hat{i}}, \Psi')$, $\mathbf{c}^\top_{\hat{f}_{\hat{i}}} := [\mathbf{c}_1^\top | \ldots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f}_{\hat{i}},\Psi'} - \underline{\Psi}_{f_{\hat{i}}}$.
      - Compute $\overline{u} \in \mathbb{Z}_p^t$ satisfying $\langle \widetilde{\mathbf{u}}, \mathbf{v}^{\mathsf{pre}}_{\hat{i}} \rangle = d^{\mathsf{pre}}_{\hat{i}} \mod p$ for $\hat{i} \in [Q']$.
      - Sample $Q - Q'$ random subsets of cardinality w using $\mathsf{SamplerSet}(N, Q, w)$, i.e. $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q'+1,Q]}$. By our setting of parameters, the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ are cover-free with an overwhelming probability.
      - For $\hat{i} \in [Q'+1, Q]$, sample $r_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_p$. Generate random shares $\{r'_k\}_{k\in[N]}$ over $\mathbb{Z}_p$ under the constraints that $\sum_{k\in\Delta_{\hat{i}}} r'_k = r_{\hat{i}}$ holds for $\hat{i} \in [Q]$, which also can be computed by the cover-freeness.

– For $k \in [N]$, set $\mathbf{u}'_k = (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, 1)^\top \in \mathbb{Z}_p^{t+2}$.

– Sample random vectors $\{\widetilde{\beta}_k\}_{k\in[N]}$ condition on the following equations:

$$\sum_{k\in\Delta_{\hat{i}}} \widetilde{\beta}_k = \mathbf{K}_{\hat{i}}^\top \cdot \begin{pmatrix}\beta_0 \\ \mathbf{c}_{\hat{f}_{\hat{i}}}\end{pmatrix} \text{ for } \hat{i} \in [Q'].$$

– Sample $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ for $k \in [N]$, Set $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}_{1,k} + \mathbf{u}'_k \mod q$.

(e) It outputs the simulated ciphertext

$$\mathsf{ct}^* \coloneqq (\Psi, \Psi', \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]}).$$

4. $\mathsf{QKeyGen}^*_{\mathsf{post}}(\mathsf{st}, f, \mathbf{v})$ generates as in the real $\mathsf{QKeyGen}$ algorithm for all 0-key queries. Otherwise, assume that the current state contains $Q'(< Q)$ tuples of $f_{\hat{i}}, \mathbf{v}_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}}, \mathbf{K}_{\hat{i}} \cdot \mathbf{v}'_{\hat{i}})$ for $f_{\hat{i}}$, for a 1-key query $(f_{\hat{i}_p}, \mathbf{v}_{\hat{i}_p})$, the simulator computes as follows.
   – Set $\Delta = \Delta_{\hat{i}_p}$ for which is chosen during QEnc^* algorithm.
   – Compute $\mathbf{P}_\Delta = \sum_{k\in\Delta} \mathbf{P}_k$ and $\widetilde{\beta}_\Delta = \sum_{k\in\Delta} \widetilde{\beta}_k$, where $\{\widetilde{\beta}_k\}_{k\in[N]}$ are chosen during QEnc^*.
   – Compute $\Psi_f \coloneqq \mathsf{HEval}_f(\Psi)$, $\mathbf{H}_{\hat{f}}$ and $\mathbf{H}_{\hat{f},\psi'}$, and use these results to compute $\mathbf{B}_{\hat{f}}$ and $\mathbf{c}_{\hat{f}}$, respectively.
   – Sample $\mathbf{J}_{\hat{i}_p} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m}, s}$, use $\mathbf{T}_{\mathbf{B}'}$ to sample $\begin{bmatrix}\mathbf{K}_{\hat{i}_p,1} \\ \mathbf{K}_{\hat{i}_p,2}\end{bmatrix}$ by $\mathsf{SampleLeft}$ such that

$$\begin{bmatrix}\mathbf{B} & \mathbf{B}_{\hat{f}_{\hat{i}_p}} \\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}_{\hat{i}_p}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{K}_{\hat{i}_p,1} \\ \mathbf{K}_{\hat{i}_p,2}\end{bmatrix} = \begin{bmatrix}\mathbf{P}_\Delta \\ \widetilde{\beta}_\Delta^\top\end{bmatrix} - \begin{bmatrix}\mathbf{B} \\ \mathbf{z}^\top\end{bmatrix} \cdot \mathbf{J}_{\hat{i}_p}.$$

   – Set $\mathbf{K}_{f_{\hat{i}_p}} = \begin{bmatrix}\mathbf{J}_{\hat{i}_p} + \mathbf{K}_{\hat{i},1} \\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$.
   – Given $d^{\mathsf{post}} = \langle \mathbf{u}^*, \mathbf{v}\rangle$, compute $\theta = d^{\mathsf{post}} - \langle \widetilde{\mathbf{u}}, \mathbf{v}\rangle$ and set $\mathbf{v}' = (\mathbf{v}, 1, \theta + r)$ for $r \coloneqq r_{\hat{i}}$.
   – Output $\mathsf{sk}_{f,\mathbf{v}} \coloneqq (\Delta, \mathbf{v}', \mathbf{K}_f \cdot \mathbf{v}')$.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572, May / June 2010.

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 467–497, December 2020.

Agr17. Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 3–35, August 2017.

AGVW13. Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 500–518, August 2013.

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326, August 2015.

ALMT20. Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 34–64, May 2020.

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362, August 2016.

AR17. Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 173–205, November 2017.

AV19. Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198, December 2019.

BGG$^+$14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556, May 2014.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273, March 2011.

BTVW17. Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302, November 2017.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

BV16. Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384, August 2016.

DDM$^+$23. Uddipana Dowerah, Subhranil Dutta, Aikaterini Mitrokotsa, Sayantan Mukherjee, and Tapas Pal. Unbounded predicate inner product functional encryption from pairings. *Journal of Cryptology*, 36(3):29, July 2023.

DGK$^+$10. Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 361–381, February 2010.

DM14. Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352, August 2014.

DOT18. Pratish Datta, Tatsuaki Okamoto, and Katsuyuki Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 640–672, December 2018.

GKP$^+$13. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

GKPV10. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 230–240. Tsinghua University Press, January 2010.

GKW16. Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388, October / November 2016.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, August 2013.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179, August 2012.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523, August 2015.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162, April 2008.

KY16. Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712, December 2016.

LLW21. Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 498–527, October 2021.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718, April 2012.

O'N10. Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Wee17. Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233, November 2017.

WFL19. Zhedong Wang, Xiong Fan, and Feng-Hao Liu. FE for inner products and its application to decentralized ABE. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 97–127, April 2019.

A Additional Preliminaries

A.1 Lattices Background

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be arbitrary matrix for some positive integers n, m, q, define the full-rank m-dimensional q-ary lattices as follows:

$$\Lambda(\mathbf{A}) = \left\{\mathbf{z} \in \mathbb{Z}^m \mid \exists\, \mathbf{s} \in \mathbb{Z}_q^n,\ s.t.\ \mathbf{A}^\top \mathbf{s} = \mathbf{z} \mod q\right\}$$
$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{z} = \mathbf{0} \mod q\}.$$

For a fixed $\mathbf{u} \in \mathbb{Z}_q^n$, define a coset of $\Lambda^\perp$ as:

$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{z} = \mathbf{u} \mod q\}.$$

The Learning with Errors problem, or LWE, is the problem of distinguishing noisy inner products from random.

Definition 3 (LWE [Reg05]) *Let $n \geq 1$ and $q \geq 2$ be integers, and let $\mathcal{X}$ be a probability distribution on $\mathbb{Z}_q$. For $\mathbf{s} \in \mathbb{Z}_q$, let $A_{\mathbf{s},\mathcal{X}}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to $\mathcal{X}$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)$.*

The decision-$\mathsf{LWE}_{q,n,\mathcal{X}}$ *problem is: for uniformly random $\mathbf{s} \in \mathbb{Z}_q$, given a* poly *number of samples that are either (all) from $A_{\mathbf{s},\mathcal{X}}$ or (all) uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, output 0 if the former holds and 1 if the latter holds.*

We say the *decision*-$\mathsf{LWE}_{q,n,\mathcal{X}}$ problem is infeasible if for all polynomial-time algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ solves the *decision*-$\mathsf{LWE}_{q,n,\mathcal{X}}$ problem (over $\mathbf{s}$ and $\mathcal{A}$'s random coins) is negligibly close to $1/2$ as a function of n.

Suppose that the error distribution $\mathcal{X}$ has a bound B, then solving $\mathsf{LWE}_{q,n,\mathcal{X}}$ is as hard as (quantumly) approximating certain worst case lattice problems to a factor of $\tilde{O}(n \cdot q/B)$. These lattice problems are hard to approximate even for subexponential q/B, i.e., 2^{n^ϵ} for some fixed $0 < \epsilon < 1/2$.

Lemma A.1 ([DM14]) *Let $\mathbf{X} \in \mathbb{R}^{n\times m}$ be a subgaussian random matrix with parameter s. There exists a universal constant $c \approx 1/\sqrt{2\pi}$ such that for any $t > 0$, we have $s_1(\mathbf{X}) \leq c \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $2/e^{\pi t^2}$.*

The following are useful facts about Gaussian distributions.

Lemma A.2 ([DGK+10], Lemma 2) *Let $\sigma > 0$, the vector $\mathbf{x} \in \mathbb{Z}^n$ be arbitrary and $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$. With overwhelming probability over the choice of $\mathbf{y}$, $|\mathbf{x}^\top \mathbf{y}| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\|$, where $\|\mathbf{y}\|_2 \leq \sigma \cdot \sqrt{n}$ and $\|\mathbf{y}\|_\infty \leq \sigma \cdot \omega(\sqrt{\log n})$.*

Lemma A.3 (Noise Flooding [GKPV10], Lemma 3) *Let $n \in \mathbb{N}$. For any real $\sigma = \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{Z}^n$,*

$$\mathsf{SD}(\mathcal{D}_{\mathbb{Z}^n,\sigma}, \mathcal{D}_{\mathbb{Z}^n,\sigma,\mathbf{c}}) \leq \|\mathbf{c}\|/\sigma$$

A.2 Multi-Output Two-Stage Sampling Algorithm

We begin by recalling several lemmas that will be useful in the subsequent proof and then present the the Multi-Output Two-Stage Sampling Algorithm and prove its security.

Lemma A.4 ([GPV08]) *Let n, m, q are integers such that $m > 2n \log q$. Then for all but an at most q^{-n} fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we have $\lambda_1^\infty(\Lambda_q(\mathbf{A})) > q/4$. Furthermore, for such $\mathbf{A}$ and any function $\omega(\sqrt{\log m})$, there is a negligible function $\varepsilon(m)$ such that $\eta_\varepsilon(\Lambda_q^\perp(\mathbf{A})) \leq \omega(\sqrt{\log m})$.*

Lemma A.5 ([LLW21]) *Let n, m, q are integers such that $m > 2n \log q$, and $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$. Then for all but an at most q^{-n} fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we have $\Lambda_1^{\infty}(\Lambda_q(\mathbf{A}|\mathbf{AR})) > q/4$. Furthermore, for such $\mathbf{A}$ and any function $\omega(\sqrt{\log m})$, there is a negligible function $\varepsilon(m)$ such that $\eta_\varepsilon(\Lambda_q^{\perp}(\mathbf{A}|\mathbf{AR})) \le \omega(\sqrt{\log m})$.*

The following algorithm is extended from the two-stage sampling algorithm 2.1 to support Q-tuples of output distributions by defining QSampler-1 and QSampler-2. Note that for the both two new-defined QSampler, each pair of the output component $(\mathbf{y}_i, \mathbf{u}_i)$ is generated independently among all $i \in [Q]$. Hence, the indistinguishability of the output by QSampler-1 and QSampler-2 can be guaranteed by Theorem 2.1.

Lemma A.6 (Multi-Output Two-Stage Sampling Algorithm) *For integers $q \ge 2$, $n \ge 1$, sufficiently large $m = O(n \log q)$, any $\mathbf{R}_i \in \mathbb{Z}^{m\times m}$, $s \ge \omega\sqrt{\log m}$ and $\rho \ge s\sqrt{m}\,\|\mathbf{R}_i\| \cdot \lambda^{\omega(1)}$ for $i \in [Q]$, the output distributions $\{(\mathbf{A}, \mathbf{AR}_i,\, \mathbf{y}_i, \mathbf{u}_i)\}_{i\in[Q]}$ of the following two procedures are statistically close.*

QSampler-1 $(\{\mathbf{R}_i\}_{i\in[Q]}, \rho, s)$: *Given matrices $\{\mathbf{R}_i\}_{i\in[Q]} \in \mathbb{Z}^{m\times m}$ and two values $\rho, s \in \mathbb{R}$ as input, this sampler performs the following steps in two stages.*

1. *Stage 1: (without the need of $\mathbf{R}_i$)*
   - *Sample a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$;*
   - *Sample random vectors $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_q^n$ for $i \in [Q]$;*
2. *Stage 2:*
   - *For each $i \in [Q]$,*
     - *Sample a random vector $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\rho}$;*
     - *Sample a vector $\mathbf{z}'_i = \begin{pmatrix}\mathbf{z}_{i,1}\\ \mathbf{z}_{i,2}\end{pmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{AR}_i, \mathbf{T}_{\mathbf{A}}, \mathbf{u}_i - \mathbf{A}\cdot\mathbf{x}_i, s)$, such that $(\mathbf{A}|\mathbf{AR}_i)\begin{pmatrix}\mathbf{z}_{i,1}\\ \mathbf{z}_{i,2}\end{pmatrix} = \mathbf{u}_i - \mathbf{A}\cdot\mathbf{x}_i \mod q$;*
     - *Set $\mathbf{y}_i = \begin{pmatrix}\mathbf{x}_i + \mathbf{z}_{i,1}\\ \mathbf{z}_{i,2}\end{pmatrix} \in \mathbb{Z}^{2m}$, satisfying $(\mathbf{A}|\mathbf{AR}_i)\mathbf{y}_i = \mathbf{u}_i \mod q$;*
   - *Output the tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i\in[Q]}$.*

QSampler-2 $(\{\mathbf{R}_i\}_{i\in[Q]}, \rho, s)$: *Given matrices $\{\mathbf{R}_i\}_{i\in[Q]} \in \mathbb{Z}^{m\times m}$ and two values $\rho, s \in \mathbb{R}$ as input, this sampler conducts the following steps in two stages.*

1. *Stage 1: (without the need of $\mathbf{R}$)*
   - *Sample a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$;*
   - *For each $i \in [Q]$, sample a random vector $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sqrt{\rho^2+s^2}}$ and set $\mathbf{u}_i = \mathbf{A}\cdot\mathbf{x}_i \mod q$;*
2. *Stage 2:*
   - *For each $i \in [Q]$,*
     - *Sample a random vector $\mathbf{z}_{i,2} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$;*
     - *Compute $\mathbf{y}_i = \begin{pmatrix}\mathbf{x}_i - \mathbf{R}\mathbf{z}_{i,2}\\ \mathbf{z}_{i,2}\end{pmatrix} \in \mathbb{Z}^{2m}$, satisfying $(\mathbf{A}|\mathbf{AR}_i)\mathbf{y}_i = \mathbf{u}_i \mod q$;*
   - *Output the tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i\in[Q]}$.*

Proof. We proceed through a series of hybrid samplers QSampler-1_q defined as follows. The first $q-1$ tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i\in[q-1]}$ are generated following the procedure of QSampler-2, the remaining $Q-q+1$ tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i\in[q;Q]}$ are generated in the manner of QSampler-1.

QSampler-1_q $(\{\mathbf{R}_i\}_{i\in[Q]}, \rho, s)$: Given matrices $\{\mathbf{R}_i\}_{i\in[Q]} \in \mathbb{Z}^{m\times m}$ and $\rho, s \in \mathbb{R}$ as input, this sampler performs the following steps in two stages.

1. Stage 1: (without the need of $\mathbf{R}_i$)
   - Sample a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$;
   - For $i \in [1; q-1]$, sample a random vector $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sqrt{\rho^2+s^2}}$ and set $\mathbf{u}_i = \mathbf{A}\cdot\mathbf{x}_i \mod q$;

– For $i \in [q; Q]$, sample random vectors $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_q^n$;

2. Stage 2:
   – For $i \in [1; q-1]$,
     - Sample a random vector $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\rho}$;
     - Sample a vector $\mathbf{z}'_i = \begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{AR}_i, \mathbf{T_A}, \mathbf{u}_i - \mathbf{A} \cdot \mathbf{x}_i, s)$, such that $(\mathbf{A}|\mathbf{AR}_i) \begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix} = \mathbf{u}_i - \mathbf{A} \cdot \mathbf{x}_i \mod q$;
     - Set $\mathbf{y}_i = \begin{pmatrix} \mathbf{x}_i + \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix} \in \mathbb{Z}^{2m}$, satisfying $(\mathbf{A}|\mathbf{AR}_i)\mathbf{y}_i = \mathbf{u}_i \mod q$;
   – For $i \in [q; Q]$,
     - Sample a random vector $\mathbf{z}_{i,2} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$;
     - Compute $\mathbf{y}_i = \begin{pmatrix} \mathbf{x}_i - \mathbf{R}\mathbf{z}_{i,2} \\ \mathbf{z}_{i,2} \end{pmatrix} \in \mathbb{Z}^{2m}$, satisfying $(\mathbf{A}|\mathbf{AR}_i)\mathbf{y}_i = \mathbf{u}_i \mod q$;
   – Output the tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i \in [Q]}$.

Note that $\mathsf{QSampler}\text{-}1_1$ is the same as $\mathsf{QSampler}\text{-}1$ and $\mathsf{QSampler}\text{-}1_{Q+1}$ is the same as $\mathsf{QSampler}\text{-}2$. Denote the output distribution of $\mathsf{QSampler}\text{-}1_q$ by $\mathcal{D}_q$. Thus, it is sufficient to prove that $\mathcal{D}_{q-1}$ is statistically close to $\mathcal{D}_q$ for $q \in [Q+1]$, implying that the output distributions $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i \in [Q]}$ of $\mathsf{QSampler}\text{-}1$ and $\mathsf{QSampler}\text{-}2$ are statistically close.

Intuitively, the indistinguishability of the output $\mathcal{D}_{q-1}$ and $\mathcal{D}_q$ can be guaranteed by 2-Stage sampling algorithm. Specifically, suppose that there exists an efficient distinguisher being able to tell $\mathcal{D}_{q-1}$ from $\mathcal{D}_q$, we can then break the indistinguishability of Theorem 2.1.

Given the challenge element $(\mathbf{A}, \mathbf{u}_q)$, namely the output of Stage 1 that generated by either $\mathsf{Sampler}$ 1 or 2, we construct $\mathsf{QSampler}\text{-}3$ taking input $\{\mathbf{R}_i\}_{i \in [Q]}$ as follows.

– For each $i \in [q-1]$,
  - Sample random vectors $\mathbf{x}'_i \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sqrt{\rho^2+s^2}}$ and $\mathbf{z}_{i,2} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$;
  - Compute $\mathbf{u}_i := (\mathbf{A}|\mathbf{AR}_i) \begin{pmatrix} \mathbf{x}'_i \\ \mathbf{z}_{i,2} \end{pmatrix} \mod q$ and denote $\mathbf{y}_i = \begin{pmatrix} \mathbf{x}'_i \\ \mathbf{z}_{i,2} \end{pmatrix}$;
– For $i = q$, query the Stage 2 oracle of $\mathsf{Sampler}$ and get response $(\mathbf{A}, \mathbf{AR}_q, \mathbf{y}_q, \mathbf{u}_q)$ satisfying $(\mathbf{A}|\mathbf{AR}_q) \cdot \mathbf{y}_q = \mathbf{u}_q$.
– For $i \in [q+1; Q]$, generate as $\mathsf{QSampler}\text{-}2$ do.
  - Sample a random vector $\mathbf{z}_{i,2} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$;
  - Compute $\mathbf{y}_i = \begin{pmatrix} \mathbf{x}_i - \mathbf{R}\mathbf{z}_{i,2} \\ \mathbf{z}_{i,2} \end{pmatrix} \in \mathbb{Z}^{2m}$, satisfying $(\mathbf{A}|\mathbf{AR}_i)\mathbf{y}_i = \mathbf{u}_i \mod q$;
– Output the tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i \in [Q]}$.

Obviously, the distribution of last $Q - q$ tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i \in [q+1;Q]}$ from above are the same as $\mathcal{D}_{q-1}$ (and $\mathcal{D}_q$).

To analyze the first $q-1$ output distribution of $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i \in [q-1]}$, we firstly consider the following two distributions:

– $\widetilde{\mathcal{D}}_1$. $\left(\mathbf{A}, \mathbf{AR}_i, \begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix}, \mathbf{u}'_i\right)_{i \in [q-1]}$: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{u}'_i \xleftarrow{\$} \mathbb{Z}_q^n$,

  $\begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix} \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{u}'_i}(\mathbf{A}|\mathbf{AR}_i), s}$ for $i \in [q-1]$.

– $\widetilde{\mathcal{D}}_2$. $\left(\mathbf{A}, \mathbf{AR}_i, \begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix}, \mathbf{u}'_i\right)_{i \in [q-1]}$: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m},s}$,

  $\mathbf{u}'_i = (\mathbf{A}|\mathbf{AR}_i) \begin{pmatrix} \mathbf{z}_{i,1} \\ \mathbf{z}_{i,2} \end{pmatrix} \mod q$ for $i \in [q-1]$.

By Lemma A.4 and A.5, for all but an at most q^{-n} fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the two distributions $\widetilde{\mathcal{D}}_1$ and $\widetilde{\mathcal{D}}_2$ are statistically close. Note that we have $\eta_\varepsilon(\lambda_q^\perp(\mathbf{A}|\mathbf{AR}_i)) \le \omega(\sqrt{\log m}) < s$. For each $i \in [q-1]$, the distribution of $(\mathbf{A}|\mathbf{AR}_i)\begin{pmatrix}\mathbf{z}_{i,1}\\ \mathbf{z}_{i,2}\end{pmatrix}$ is uniformly random $\mathbb{Z}_q^n$, and the conditional distribution of $\begin{pmatrix}\mathbf{z}_{i,1}\\ \mathbf{z}_{i,2}\end{pmatrix}$ under the constraint is $\mathcal{D}_{\Lambda_q^{\mathbf{u}'_i}(\mathbf{A}|\mathbf{AR}_i),s}$. Due to the reason that each tuple for $i \in [q-1]$ is generated independently, the joint distribution $\widetilde{\mathcal{D}}_1 \overset{s}{\approx} \widetilde{\mathcal{D}}_2$.

Next, we decompose the component $\mathbf{x}'_i$ sampled in the Stage 1 of QSampler-3 into $\mathbf{x}_i + \mathbf{z}_{i,1}$ (within a negligible statistical distance), where $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\rho}$, $\mathbf{z}_{i,1} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$. The decomposition holds as we have $\rho > s > \eta_\varepsilon(\mathbb{Z}^m)$ for some $\varepsilon = \mathsf{negl}(\lambda)$. Again, each component for $i \in [q-1]$ is generated independently, the above indistinguishability thus immediately implies the indistinguishability of the following two distributions:

- $\widetilde{\mathcal{D}}'_1$. $\left(\mathbf{A}, \mathbf{AR}_i, \begin{pmatrix}\mathbf{z}_{i,1}+\mathbf{x}_i\\ \mathbf{z}_{i,2}\end{pmatrix}, \mathbf{u}'_i + \mathbf{Ax}_i\right)_{i\in[q-1]}$: $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\rho}$, the other random variables are sampled as $\widetilde{\mathcal{D}}_1$.
- $\widetilde{\mathcal{D}}'_2$. $\left(\mathbf{A}, \mathbf{AR}_i, \begin{pmatrix}\mathbf{z}_{i,1}+\mathbf{x}_i\\ \mathbf{z}_{i,2}\end{pmatrix}, \mathbf{u}'_i + \mathbf{Ax}_i\right)_{i\in[q-1]}$: $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\rho}$, the other random variables are sampled as $\widetilde{\mathcal{D}}_2$.

By replacing the variable $\mathbf{u}_i$ of $\mathbf{u}'_i+\mathbf{Ax}_i$, the marginal distribution of $\{\mathbf{u}_i\}$ is found to be still uniformly random in $\widetilde{\mathcal{D}}'_1$. Then it is not hard to see that $\widetilde{\mathcal{D}}'_1$ is distributed identical as the tuples $\{(\mathbf{A}, \mathbf{AR}_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i\in[q-1]}$ output by the defined both QSampler-1_{q-1} and QSampler-1_q, $\widetilde{\mathcal{D}}'_2$ is distributed statistically close to that output by QSampler-3.

Hence, QSampler-3 successfully simulates all the output of QSampler-1_{q-1} and QSampler-1_q except the q-th tuple $(\mathbf{A}, \mathbf{AR}_q, \mathbf{y}_q, \mathbf{u}_q)$. If the response $(\mathbf{A}, \mathbf{AR}_q, \mathbf{y}_q, \mathbf{u}_q)$ returned by the Stage 2 oracle of Sampler is computed using Sampler-1, then QSampler-3 simulates $\mathcal{D}_{q-1}$, otherwise QSampler-3 simulates $\mathcal{D}_q$. In other words, $\mathcal{D}_{q-1} \overset{s}{\approx} \mathcal{D}_q$ for $q \in [Q+1]$. This completes the proof.

□

□

B Supplementary Material of Section 3

In this section, we provide the parameters setting and security proofs for the predicate encryption schemes described in Section 3, which were omitted from the main text due to space limitations.

B.1 Supplementary Material of (1, poly) PE Scheme in Section 3.1

Parameter Setting. We choose the parameters so that correctness and security of the scheme are satisfied. We must satisfy the following constraints.

1. For correctness, the final magnitude of error obtained must be below $q/4$. Let us recall the decryption procedure and analyze the noise component $\mathbf{e}_{\mathsf{Eval}}$ causing by homomorphic evaluations.

$$\begin{aligned}
\mathbf{c}_{\hat{f}}^\top &:= [\mathbf{c}_1^\top|\dots|\mathbf{c}_L^\top]\cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f \\
&= \mathbf{s}^\top[\mathbf{B}_1 + \psi_1\overline{\mathbf{G}}|\dots|\mathbf{B}_L + \psi_L\overline{\mathbf{G}}]\cdot \mathbf{H}_{\hat{f},\psi} + \mathbf{e}^\top \underbrace{[\mathbf{W}_1|\dots|\mathbf{W}_L]\cdot \mathbf{H}_{\hat{f},\psi}}_{\mathbf{W}_{\hat{f}}} - \underline{\Psi}_f \\
&= \mathbf{s}^\top(\mathbf{B}_{\hat{f}} + \overline{\Psi}_f) - \underline{\Psi}_f + \underbrace{\mathbf{e}^\top\mathbf{W}_{\hat{f}}}_{\mathbf{e}_{\mathsf{attr.Eval}}} \\
&= \mathbf{s}^\top\mathbf{B}_{\hat{f}} + f(\mathbf{x})\cdot[\mathbf{s}^\top|-1]\cdot\mathbf{G} + \mathbf{e}_{\mathsf{attr.Eval}} - \underbrace{\mathbf{e}^\top\mathbf{R}_f}_{\mathbf{e}_{\mathsf{HE.Eval}}}
\end{aligned}$$

Thus, $\mathbf{e}_{\mathsf{Eval}} = \mathbf{e}_{\mathsf{attr.Eval}} + \mathbf{e}_{\mathsf{HE.Eval}} = (\mathbf{W}_{\hat{f}} - \mathbf{R}_f)^\top \mathbf{e}$, where $\mathbf{W}_{\hat{f}}$ is bound by Lemma 2.2, $\mathbf{R}_f$ is the randomness of FHE ciphertext Ψ_f and is then bound by Lemma 2.8. As a result, $\mathbf{e}_{\mathsf{dec}} = \mathbf{e}' - \mathbf{K}_f^\top \begin{pmatrix} \mathbf{e} \\ (\mathbf{W}_{\hat{f}} - \mathbf{R}_f)^\top \mathbf{e} \end{pmatrix}$.

2. We must choose m large enough for the algorithm $\mathsf{TrapGen}$ (Lemma 2.1).
3. We must choose s_B such that LWE_{q,n,s_B} assumption holds.
4. We set s used in $\mathsf{SampleLeft}$ (Lemma 2.2) and $\mathsf{SampleRight}$ (Lemma 2.3) such that the output matrices are statistically indistinguishable.
5. We set s and ρ to meet the requirements of two-stage sampling techniques (Theorem 2.1).
6. We must choose the parameter s_D used to sample the error $\mathbf{e}'$ in κ large enough so that the following equations are both satisfied:

$$\mathbf{e}' \overset{s}{\approx} \mathbf{J}^{*\top} \cdot \mathbf{e} + \mathbf{e}',$$

where $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m}, \sqrt{\rho^2+s^2}}$.

Similarly to [Agr17, BTVW17, LLW21], we choose our parameters to satisfy these constraints. Our parameters may be chosen as: $n = \mathsf{poly}(\lambda)$, $m = (n+1)\log q$, $s_B = O(\sqrt{n})$, $s = O(Ln \log q)^{O(\hat{d})} \cdot \omega(\sqrt{\log m})$, $\rho = O(Ln \log q)^{O(\hat{d})} \cdot \omega(\sqrt{\log m}) \cdot \lambda^{\omega(1)}$, $s_D = \sqrt{n} \cdot m \cdot \rho \cdot \lambda^{\omega(1)}$, $q = 4m\sqrt{nm} \cdot \rho \cdot \lambda^{\omega(1)}$, where $L = \ell \cdot (n+1)^2 \log q^2$, $\hat{d} = d \cdot O(\log m \log \log q)$.

Theorem (Restatement of Theorem 3.1) *Assuming the hardness of* LWE*, then the construction 1 is a* PE *for the class* $\mathcal{F}$*, achieving* $(1, \mathsf{poly})$*-sel-SIM security that allows at most single 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

Proof. We define a PPT simulator Sim and prove that for any PPT adversary $\mathcal{A}$, the ideal experiment with respect to Sim is computationally indistinguishable (under the LWE assumption) from the output of the real experiment.

Simulator. $\mathsf{Sim}(1^\lambda, 1^{|\mathbf{x}|})$:

1. $\mathsf{Setup}^*(1^\lambda, 1^{|\mathbf{x}|})$ generates all public parameters as in the real Setup and initializes $\mathsf{st} := \emptyset$.
2. $\mathsf{KeyGen}^*_{\mathsf{pre}}(\mathsf{st}, f)$ generates all public parameters as in the real KeyGen and maintains st that contains (f, sk_f) if the adversary queried for 1-key such that $f(\mathbf{x}^*) = 0$.
3. $\mathsf{Enc}^*(\mathsf{st})$ takes as the state value st and constructs the challenge ciphertext as follows.
   - Sample $\beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ independently and uniformly from $\mathbb{Z}_q^m$.
   - Sample $\{\Psi_i\}_{i\in[\ell]}$ uniformly from $\mathbb{Z}_q^{(n+1)\times(n+1)\log q}$.
   - If $\mathsf{st} = \emptyset$, then sample κ randomly from $\mathbb{Z}_q^m$.
   - Otherwise, if $\mathsf{st} = (f, \mathsf{sk}_f = \mathbf{K}_f, \mu)$, which means that the adversary has made a pre-challenge 1-key query for f. Then, Enc^* generates κ to satisfy the decryption consistency as follows.
     - Compute $\Psi_f := \mathsf{HEval}_f(\Psi)$ and $\mathbf{H}_{\hat{f},\psi} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}, \psi)$.
     - Compute $\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top | \ldots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f$.
     - Set $\kappa = \mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} + \mathbf{e}' + \mathbf{b}$ for $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ and $\mathbf{b} = [0, \ldots, 0, \lceil q/2 \rceil \mu]^\top$.
   - It outputs the simulated ciphertext

$$\mathsf{ct}^* := (\Psi, \beta_0, \kappa, \{\mathbf{c}_j\}_{j\in[L]})$$

4. $\mathsf{KeyGen}^*_{\mathsf{post}}(\mathsf{st}, f)$: It generates all secret keys as in the real KeyGen.

Auxiliary Algorithms.

$\mathsf{Setup}_1^*(1^\lambda, \mathbf{x}^*)$: Do the following:

1. Sample $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
2. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, compute $\mathbf{c}^\top := \mathbf{s}^\top \mathbf{B} + \mathbf{e}^\top$.
3. Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m \times m}$ for $i \in [\ell]$ and compute

$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G}.$$

   Let $\psi = (\psi_1, \ldots, \psi_L)$ denote the bit-representation of $\Psi := [\Psi_1 | \cdots | \Psi_\ell]$.
4. Let $\mathbf{B}_j = \mathbf{B} \cdot \mathbf{W}_j - \psi_j \cdot \overline{\mathbf{G}}$ for $j \in [L]$, where $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m \times m}$ for $j \in [L]$.
5. Sample $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m}, \sqrt{\rho^2+s^2}}$ and set $\mathbf{P} = \mathbf{B} \cdot \mathbf{J}^* \mod q$.
6. Initialize $\mathsf{st} := \emptyset$. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P}),\ \mathsf{msk} := (\mathbf{T_B}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{W}_j\}_{j\in[L]}, \mathbf{J}^*, \mathbf{s}).$$

$\mathsf{Enc}_1^*(\mathsf{mpk}, \mathsf{msk}, \mu)$: Do the following:

1. Set $\beta_0 := \mathbf{c}$.
2. Sample $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$, set $\mathbf{b} = [0, \ldots, 0, \lceil q/2 \rceil \mu]^\top \in \mathbb{Z}_q^m$ and compute $\kappa := \mathbf{P}^\top \mathbf{s} + \mathbf{e}' + \mathbf{b}$.
3. For $j \in [L]$, compute $\mathbf{c}_j := \mathbf{W}_j^\top \beta_0$, where $\mathbf{W}_j$ are the matrices in the msk generated by Setup_1^*.
4. Output the ciphertext $\mathsf{ct}^* := (\Psi, \beta_0, \kappa, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{KeyGen}_1^*(\mathsf{msk}, \mathsf{st}, f)$: Do the following:

1. Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute

$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}).$$

$$\begin{aligned} \mathbf{B}_{\hat{f}} &:= [\mathbf{B}_1 \mid \cdots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}} \\ &= [\mathbf{B}_1 + \psi_1 \overline{\mathbf{G}} \mid \cdots \mid \mathbf{B}_L + \psi_L \overline{\mathbf{G}}] \cdot \mathbf{H}_{\hat{f},\psi} - \overline{\Psi}_f \\ &= \mathbf{B}[\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi} - \overline{\Psi}_f \\ &= \mathbf{B}(\mathbf{W}_{\hat{f}} - \mathbf{R}_f) - f(\mathbf{x}^*)\overline{\mathbf{G}} \end{aligned}$$

   where

$$\mathbf{W}_{\hat{f}} := [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi},$$
$$\Psi_f = \begin{pmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{pmatrix} \mathbf{R}_f + f(\mathbf{x}^*) \begin{pmatrix} \overline{\mathbf{G}} \\ \underline{\mathbf{G}} \end{pmatrix} = \begin{pmatrix} \overline{\Psi}_f \\ \underline{\Psi}_f \end{pmatrix} = \begin{pmatrix} \mathbf{B}\mathbf{R}_f + f(\mathbf{x}^*)\overline{\mathbf{G}} \\ \mathbf{c}^\top \mathbf{R}_f + f(\mathbf{x}^*)\underline{\mathbf{G}} \end{pmatrix}.$$

2. For 0-key query such that $f(\mathbf{x}^*) \neq 0$, generate

$$\mathbf{K}_f \leftarrow \mathsf{SampleRight}(\mathbf{B}, \mathbf{G}, \mathbf{W}_{\hat{f}} - \mathbf{R}_f, \mathbf{P}, s).$$

3. For 1-key query such that $f(\mathbf{x}^*) = 0$, sample $\mathbf{K}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m}, s}$, set sk_f as $\mathbf{K}_f$ and update st to contain (f, sk_f).

$$\mathbf{K}_f := \begin{bmatrix} \mathbf{J}^* - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f) \cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix}.$$

4. Output $\mathsf{sk}_f := \mathbf{K}_f$.

$\mathsf{Enc}_2^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Do the following:

1. Generate $\beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ as in Enc_1^*.
2. Sample $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$, set $\mathbf{b} = [0, \dots, 0, \lceil q/2 \rceil \mu]^\top \in \mathbb{Z}_q^m$ and compute κ as follows:
   - If there is no pre-challenge 1-key query, then it computes $\kappa = \mathbf{J}^{*\top} \cdot \beta_0 + \mathbf{e}' + \mathbf{b}$.
   - If the adversary has already queried the 1-key for f, then it computes $\kappa = (\mathsf{sk}_f)^\top \cdot \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} + \mathbf{e}' + \mathbf{b}$, where $\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top | \dots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f$.
3. Output the ciphertext $\mathsf{ct}^* := (\Psi, \beta_0, \kappa, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{Setup}_2^*(1^\lambda, \mathbf{x}^*)$: Sample $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_q^m$. Compute and set the remaining components as in Setup_1^*.
$\mathsf{KeyGen}_2^*(\mathsf{msk}, \mathsf{st}, f)$: Do the following:

1. For 0-key query, generate the key $\mathbf{K}_f$ as in KeyGen.
2. For 1-key query, set the key $\mathbf{K}_f$ as in KeyGen_1^*.
3. Output $\mathsf{sk}_f := \mathbf{K}_f$.

$\mathsf{Enc}_3^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: If there is no pre-challenge 1-key, it samples κ randomly from $\mathbb{Z}_q^m$. Otherwise, compute the ciphertext as in Enc_2^*.
$\mathsf{Enc}_4^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Sample $\mathbf{c}_j$ uniformly. Compute the remaining ciphertext elements as in Enc_3^*.
$\mathsf{Enc}_5^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$: Do the following:

1. Sample $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_q^m$ and set $\beta_0, \mathbf{c}_j, \kappa$ as in Enc_4^*.
2. Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times(n+1)\log q}$ for $i \in [\ell]$ and compute

$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G}.$$

3. Output the ciphertext $\mathsf{ct}^* := (\Psi, \beta_0, \kappa, \{\mathbf{c}_j\}_{j\in[L]})$.

Hybrids.

$\mathcal{H}_0$: The real experiment.
$\mathcal{H}_1$: The real game algorithms Setup and Enc are replaced with Setup_1^* and Enc_1^*, which use the knowledge of $\mathbf{x}^*$ to generate the public parameters, the master public/secret keys, and additionally sets $\mathbf{P} = \mathbf{B} \cdot \mathbf{J}^*$. $\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically close by an application of the Leftover Hash Lemma.
$\mathcal{H}_2$: The real game algorithm KeyGen is replaced with KeyGen_1^* where instead of using the trapdoor $\mathbf{T}_\mathbf{B}$ of the matrix $\mathbf{B}$, secret keys for a 0-key queries are sampled using the public trapdoor $\mathbf{T}_\mathbf{G}$ along with the trapdoor information generated in Setup_1^*, and the secret key for the 1-key query for function f is computed as $\mathbf{K}_f := \begin{bmatrix} \mathbf{J}^* - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f) \cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix}$.
$\mathcal{H}_3$: Enc_1^* is replaced by Enc_2^*, in which κ is computed from $\mathbf{J}^*$ if there is no 1-key queried before, or otherwise from the 1-key computed by KeyGen_1^*.
$\mathcal{H}_4$: Setup_1^* is replaced by Setup_2^*, in which $\mathbf{B}$ and $\mathbf{c}$ are chosen randomly and thus all public matrices and ciphertext elements are derived from $(\mathbf{B}, \mathbf{c})$.

$\mathcal{H}_5$: KeyGen_1^* is replaced by KeyGen_2^*. The algorithm KeyGen_2^* is as the same as KeyGen_1^* except for the response to the 1-key query.
$\mathcal{H}_6$: Enc_2^* is replaced by Enc_3^*. The difference between Enc_2^* and Enc_3^* is that when there is no pre-challenge 1-key query, then in the former, κ is computed from $\mathbf{J}^*$ whereas in the latter, κ is chosen randomly.
$\mathcal{H}_7$: The key generation algorithm is changed from KeyGen_2^* to KeyGen^* algorithm.
$\mathcal{H}_8$: The encryption algorithm is changed from Enc_3^* to Enc_4^*, in which the ciphertext elements $\mathbf{c}_j$ are switched to random.
$\mathcal{H}_9$: The algorithms Setup_2^* and Enc_4^* are replaced with the real Setup and Enc_5^*. This hybrid is identical to the ideal experiment except that the FHE ciphertexts are computed during encryption.
$\mathcal{H}_{10}$: Enc_5^* is replaced by Enc^*. This hybrid is identical to the ideal experiment when running the simulator Sim.

Next, we will prove that each pair of adjacent hybrid arguments is indistinguishable.

Lemma B.1 *$\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is in how the public parameters and the ciphertext are generated.

1. The public parameters
   The matrix $\mathbf{B}$ in both two hybrids is generated using $\mathsf{TrapGen}$ algorithm and hence distributed close to uniform by Lemma 2.1. The difference of public parameters between the two hybrids is how the remaining public parameters are generated. In $\mathcal{H}_0$, public matrices $\{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P}$ are chosen uniformly and independently. In $\mathcal{H}_1$, we set $\mathbf{B}_j = \mathbf{B}\mathbf{W}_j - \psi_j\overline{\mathbf{G}}, \mathbf{P} := \mathbf{B}\mathbf{J}^*$ for $j \in [L]$ for $\mathbf{W}_j \xleftarrow{\$} \{0,1\}^{m\times m}$, $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$.
   By Leftover Hash Lemma (Lemma 2.6), we have
   $$(\mathbf{B}, \mathbf{B}\mathbf{W}_j - \psi_j\overline{\mathbf{G}}, \mathbf{W}_j^\top \mathbf{e}) \overset{s}{\approx} (\mathbf{B}, \mathcal{U}, \mathbf{W}_j^\top \mathbf{e}),$$
   where $\mathcal{U}$ denote the uniform distribution over $\mathbb{Z}_q^{n\times m}$.
   By Lemma 2.4, we have
   $$(\mathbf{B}, \mathbf{B}\cdot\mathbf{J}^*) \overset{s}{\approx} (\mathbf{B}, \mathcal{U}(\mathbb{Z}_q^{n\times m})).$$
   Therefore, the distribution of the public parameters $\{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P}$ in $\mathcal{H}_1$ is statistically close to that in $\mathcal{H}_0$.
2. The ciphertext
   The difference of the ciphertext between the two hybrids is in how the ciphertext elements $\mathbf{c}_j$ are generated. In $\mathcal{H}_0$, for $j \in [L]$, we have
   $$\mathbf{c}_j := [\mathbf{B}_j + \psi_j\overline{\mathbf{G}}]^\top \mathbf{s} + \mathbf{W}_j^\top \mathbf{e}$$
   In $\mathcal{H}_1$, for $j \in [L]$, since $\beta_0^\top = \mathbf{s}^\top\mathbf{B} + \mathbf{e}^\top$, we can rewrite $\mathbf{c}_j = \mathbf{W}_j^\top \beta_0$ as:
   $$\begin{aligned}\mathbf{c}_j &= \mathbf{W}_j^\top(\mathbf{B}^\top \cdot \mathbf{s} + \mathbf{e})\\ &= (\mathbf{B}\mathbf{W}_j)^\top \cdot \mathbf{s} + \mathbf{e}\\ &= (\mathbf{B}_j + \psi_j\overline{\mathbf{G}})^\top \cdot \mathbf{s} + \mathbf{W}_j^\top \mathbf{e} \text{ (as in } \mathcal{H}_0).\end{aligned}$$

Hence, the joint distribution of the public parameters and ciphertext is statistically indistinguishable between the two hybrids. □

□

Lemma B.2 *$\mathcal{H}_1$ and $\mathcal{H}_2$ are statistically indistinguishable.*

Proof. In both $\mathcal{H}_1$ and $\mathcal{H}_2$, due to the reason that in Setup_1^* the attribute encoding matrices $\mathbf{B}_j$ are programmed as $\mathbf{BW}_j - \psi_j\overline{\mathbf{G}}$, the encoding matrix $\mathbf{B}_{\hat{f}}$ after homomorphic evaluation is in the exact same form $\mathbf{B}(\mathbf{W}_{\hat{f}} - \mathbf{R}_f) - f(\mathbf{x}^*)\overline{\mathbf{G}}$ as described in KeyGen_1^*.

The difference between the two hybrids is in the way the queried secret keys are generated. We consider the following two cases:

1. For the 0-key query of f, in $\mathcal{H}_1$, these keys are sampled using the $\mathsf{SampleLeft}$ algorithm, whereas in $\mathcal{H}_2$, they are sampled using the $\mathsf{SampleRight}$ algorithm. By employing the lemma 2.3, the resulting distributions are statistically indistinguishable.
2. For the 1-key query of f, in $\mathcal{H}_1$, the secret key is sampled using the Sample-1, while in $\mathcal{H}_2$, the secret key is sampled using the Sample-2. Thus, by Theorem 2.1, the two hybrids are statistically indistinguishable.

□

□

Lemma B.3 *$\mathcal{H}_2$ and $\mathcal{H}_3$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way how the ciphertext element κ is generated.

In $\mathcal{H}_2$, $\kappa = \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{e}' + \mathbf{b}$. In $\mathcal{H}_3$, we consider the following two cases:

1. If there is no pre-challenge 1-key query, then
$$\begin{aligned}\kappa &= \mathbf{J}^{*\top} \cdot \beta_0 + \mathbf{e}' + \mathbf{b}\\ &= \mathbf{BJ}^\top \cdot \mathbf{s} + \mathbf{J}^{*\top} \cdot \mathbf{e} + \mathbf{e}' + \mathbf{b}\\ &= \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{J}^{*\top} \cdot \mathbf{e} + \mathbf{e}' + \mathbf{b}\end{aligned}$$
Thus, it suffices to ensure that
$$\mathbf{e}' \overset{s}{\approx} \mathbf{J}^{*\top} \cdot \mathbf{e} + \mathbf{e}'.$$
By the noise flooding (Lemma A.3) and our setting of parameters, the above equation is satisfied.
2. If the adversary has already queried the 1-key for f, we have
$$\kappa = (\mathsf{sk}_f)^\top \cdot \begin{pmatrix}\beta_0\\ \mathbf{c}_{\hat{f}}\end{pmatrix} + \mathbf{e}' + \mathbf{b}.$$
Recall that
$$\begin{aligned}\mathbf{c}_{\hat{f}}^\top &:= [\mathbf{c}_1^\top | \dots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f\\ &= \beta_0^\top [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi} - \underline{\Psi}_f\\ &= \beta_0^\top \mathbf{W}_{\hat{f}} - \mathbf{c}^\top \mathbf{R}_f\end{aligned}$$
Thus,
$$\begin{aligned}\kappa &:= \mathbf{K}_f^\top \cdot \begin{pmatrix}\beta_0\\ \mathbf{c}_{\hat{f}}\end{pmatrix} + \mathbf{e}_1 + \mathbf{b}\\ &= \begin{bmatrix}\mathbf{J}^* - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f) \cdot \mathbf{K}_2\\ \mathbf{K}_2\end{bmatrix}^\top \begin{pmatrix}\beta_0\\ (\mathbf{W}_{\hat{f}} - \mathbf{R}_f)^\top \beta_0\end{pmatrix} + \mathbf{e}' + \mathbf{b}\\ &= \mathbf{J}^{*\top}\beta_0 + \mathbf{e}' + \mathbf{b},\end{aligned}$$
as in the first case.

□

□

Lemma B.4 *$\mathcal{H}_3$ and $\mathcal{H}_4$ are computationally indistinguishable under the* LWE *assumption.*

Proof. We show how the LWE assumption can be broken given an adversary that distinguishes between Hybrid 3 and Hybrid 4. Given the LWE challenge sample $(\mathbf{B}, \mathbf{c})$ where $\mathbf{c}$ is either real or random. The reduction does as follows:

1. Run Setup_2^* given the instance $(\mathbf{B}, \mathbf{c})$. We note that the generation of public parameters can be implemented without the trapdoor of $\mathbf{B}$.
2. Run KeyGen_1^* and Enc_2^* accordingly.

Note that if $\mathbf{c} = \mathbf{B}^\top \mathbf{s} + \mathbf{e}$, then we simulate the distribution of $\mathcal{H}_3$, while we simulate $\mathcal{H}_4$ if $\mathbf{c}$ is random. □

□

Lemma B.5 *$\mathcal{H}_4$ and $\mathcal{H}_5$ are statistically indistinguishable.*

Proof. The proof is analogous to the proof of indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ for generating secret keys of 0-key queries. □

□

Lemma B.6 *$\mathcal{H}_5$ and $\mathcal{H}_6$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way of generating ciphertext element κ in the case that there is no pre-challenge 1-key queried.

In $\mathcal{H}_5$, if there is no 1-key queried,

$$\kappa = \mathbf{J}^{*\top} \cdot \beta_0 + \mathbf{e}' + \mathbf{b}$$

Since $\beta_0 := \mathbf{c} \xleftarrow{\$} \mathbb{Z}_q^m$, then $\mathbf{J}^{*\top} \cdot \beta_0 \overset{s}{\approx} \mathcal{U}$ by Lemma 2.6. Thus, $\kappa \overset{s}{\approx} \mathcal{U}$, as is in $\mathcal{H}_6$. □

□

Lemma B.7 *$\mathcal{H}_6$ and $\mathcal{H}_7$ are statistically indistinguishable.*

Proof. The proof is analogous to the proof of indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ for generating secret keys of 1-key queries. □

□

Lemma B.8 *$\mathcal{H}_7$ and $\mathcal{H}_8$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way to generate ciphertext elements $\mathbf{c}_j$.

In $\mathcal{H}_7$, the ciphertext elements are set as $\mathbf{c}_j = \mathbf{W}_j^\top \beta_0$. While, in $\mathcal{H}_8$, $\mathbf{c}_j$ are chosen independently and randomly from $\mathbb{Z}_q^m$.

Recall that $\beta_0 \overset{s}{\approx} \mathcal{U}$ in both hybrids, thus the indistinguishability of two hybrids follows from the Leftover Hash Lemma (Lemma 2.6):

$$(\mathbf{B}, \beta_0, \{\mathbf{B}\mathbf{W}_j, \mathbf{W}_j^\top \beta_0\}) \overset{s}{\approx} (\mathbf{B}, \beta_0, \{\mathcal{U}, \mathcal{U}\}).$$

□

□

Lemma B.9 *$\mathcal{H}_8$ and $\mathcal{H}_9$ are statistically indistinguishable.*

Proof. The proof follows similarly as the proof of indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$. □

□

Lemma B.10 *$\mathcal{H}_9$ and $\mathcal{H}_{10}$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is how the FHE ciphertext elements Ψ_i are generated.

In $\mathcal{H}_9$, the FHE ciphertext elements Ψ_i are computed by

$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G}$$

where $\mathbf{R}_i$ is randomness chosen independently from $\{0,1\}^{m \times m}$. By Leftover Hash Lemma (Lemma 2.6), we have that $\begin{pmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{pmatrix} \mathbf{R}_i$ is statistically close to uniform.

Therefore, $\Psi_i \overset{s}{\approx} \mathcal{U}$, as in $\mathcal{H}_{10}$ (ideal experiment). □

□

□

□

B.2 Supplementary Material of (Q, poly) PE Scheme in Section 3.2

Parameter Setting. We choose the parameters so that correctness and security of the scheme are satisfied. We must satisfy the following constraints.

1. For correctness, the final magnitude of error obtained must be below $q/4$.
2. We must choose m large enough for the algorithm $\mathsf{TrapGen}$ (Lemma 2.1).
3. We must choose s_B such that LWE_{q,n,s_B} assumption holds.
4. We set s used in $\mathsf{SampleLeft}$ (Lemma 2.2) and $\mathsf{SampleRight}$ (Lemma 2.3) such that the output matrices are statistically indistinguishable.
5. We set s and ρ to meet the requirements of two-stage sampling techniques (Theorem 2.1).
6. We must choose the parameter s_D used to sample the error $\mathbf{e}'_k$ in β_k large enough so that the following equations are satisfied for $k \in [N]$:

$$\mathbf{e}'_k \overset{s}{\approx} \mathbf{J}_k^\top \cdot \mathbf{e} + \mathbf{e}'_k,$$

   where $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, $\mathbf{J}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m}, s}$ for $k \in \{\delta_1, \ldots, \delta_Q\}$ and $\mathbf{J}_k \overset{s}{\approx} \mathcal{D}_{\mathbb{Z}^{m\times m}, \sqrt{\rho^2+s^2}}$ for $k \in [N] \backslash \{\delta_1, \ldots, \delta_Q\}$.
7. We must choose N, w, Q for cover-freeness (Lemma 2.7).

Similarly to the parameter choices for (1,poly)-PE scheme (Construction 1), we choose our parameters to satisfy these constraints. Our parameters may be chosen as: $n = \mathsf{poly}(\lambda)$, $m = (n+1)\log q$, $Q = O(\lambda)$, $w = \Theta(\lambda)$, $N = O(w\lambda^3)$, $s_B = O(\sqrt{n})$, $s = O(Ln\log q)^{O(\hat{d})} \cdot \omega(\sqrt{\log m})$, $\rho = O(Ln\log q)^{O(\hat{d})} \cdot \omega(\sqrt{\log m}) \cdot \lambda^{\omega(1)}$, $s_D = \sqrt{n} \cdot m \cdot \rho \cdot \lambda^{\omega(1)}$, $q = 4m\sqrt{wnm} \cdot \rho \cdot \lambda^{\omega(1)}$, where $L = \ell \cdot (n+1)^2 \log q^2$, $\hat{d} = d \cdot O(\log m \log\log q)$.

Theorem (Restatement of Theorem 3.2) *Assuming the hardness of* LWE*, then the construction 2 is a* PE *for the class* $\mathcal{F}$*, achieving* (Q, poly)-sel-*SIM security that allows up to Q 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

Proof. We define a PPT simulator Sim and prove that for any PPT adversary $\mathcal{A}$, the ideal experiment with respect to Sim is computationally indistinguishable (under the LWE assumption) from the output of the real experiment.

Simulator. $\mathsf{Sim}(1^\lambda, 1^{|\mathbf{x}|})$:

1. $\mathsf{QSetup}^*(1^\lambda, 1^{|\mathbf{x}|})$ generates all public parameters as in the real $\mathsf{QPE.Setup}$, except that it runs $(\mathbf{B}', \mathbf{T}_{\mathbf{B}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^m, q)$, then parses $\mathbf{B}' = \begin{bmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{bmatrix}$, where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$. Set $\mathbf{B}$ as the public matrix in mpk and initialize $\mathsf{st} := \emptyset$.

2. $\mathsf{QKeyGen}^*_{\mathsf{pre}}(\mathsf{st}.f)$ generates all secret keys as in the real $\mathsf{QPE.KeyGen}$ and maintains st to contain $\{f_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{K}_{\hat{i}})\}$ for 1-key query that $f_{\hat{i}}(\mathbf{x}^*) = 0$.
3. $\mathsf{QEnc}^*(\mathsf{st})$ takes as input the stateful value st and construct the challenge ciphertext as follows.
   - Sample $\{\mathbf{c}_j\}_{j\in[L]}$ independently and uniformly from $\mathbb{Z}_q^m$, and sets $\beta_0 = \mathbf{z}$, where $\mathbf{z}$ is prepared during QSetup^*.
   - Sample $\{\Psi_i\}_{i\in[\ell]}$ uniformly from $\mathbb{Z}_q^{(n+1)\times(n+1)\log q}$. Let $\psi = (\psi_1, \ldots, \psi_L)$ denote the bit-representation of $\Psi := [\Psi_1|\cdots|\Psi_\ell]$.
   - If $\mathsf{st} = \emptyset$, i.e., the adversary did not make any 1-key pre-challenge query, it computes $\{\beta_{1,k}\}_{k\in[N]}$ as follows:
     - Choose $2Q$ random subset $(\Delta_1, \ldots, \Delta_Q)$ and $(\Delta'_1, \ldots, \Delta'_Q)$ with size w according sampler $\mathsf{SamplerSet}$ $(N, 2Q, w)$.
     - Generate random shares $\{b_k\}_{k\in[N]}$ over $\mathbb{Z}_q$ under the following constraints: for $\hat{i} \in [Q]$, $\sum_{k\in\Delta_{\hat{i}}} b_k = 0$, $\sum_{k\in\Delta'_{\hat{i}}} b_k = \lceil q/2 \rceil$. This can be done efficiently by the cover-freeness of the subsets, using the following standard procedure.
     Let $\delta_{\hat{i}}$ and $\delta'_{\hat{i}}$ be the unique index of $\Delta_{\hat{i}}$ and $\Delta'_{\hat{i}}$, respectively. To generate random shares $\{b_k\}_{k\in[N]}$, we first sample b_k randomly for all $k \in [N]\backslash\{\delta_{\hat{i}}, \delta'_{\hat{i}}\}_{\hat{i}\in[Q]}$, and then fix $b_{\delta_{\hat{i}}} = -\sum_{k\in\Delta_{\hat{i}}\backslash\{\delta_{\hat{i}}\}} b_k$ and $b_{\delta'_{\hat{i}}} = \lceil q/2 \rceil - \sum_{k\in\Delta'_{\hat{i}}\backslash\{\delta'_{\hat{i}}\}} b_k$ for $\hat{i} \in [Q]$.
     - Set $\mathbf{b}_k = [0, \ldots, 0, b_k]^\top \in \mathbb{Z}_q^m$, sample $\widetilde{\beta}_k \xleftarrow{\$} \mathbb{Z}_q^m$ and $\mathbf{e}'_k \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ for $k \in [N]$.
     - Set $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}'_k + \mathbf{b}_k \mod q$.
   - If $\mathsf{st} = (\{f_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{K}_{\hat{i}})\}_{\hat{i}\in[Q']}, \mu)$, which means that the adversary has already made Q' 1-key queries. Then, QEnc^* generates $\{\beta_{1,k}\}_{k\in[N]}$ to satisfy the decryption consistency as follows.
     - Compute $\Psi_{f_{\hat{i}}} := \mathsf{HEval}_{f_{\hat{i}}}(\Psi)$ and $\mathbf{H}_{\hat{f}_{\hat{i}},\Psi} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}_{\hat{i}}, \Psi)$ for $\hat{i} \in [Q']$.
     - Sample $Q - Q'$ random subsets of cardinality w according sampler $\mathsf{SamplerSet}(N, Q, w)$, i.e., $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q'+1,Q]}$. By our setting of parameters, the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ are cover-free with an overwhelming probability.
     - Sample random shares $\{b_k\}_{k\in[N]}$ over $\mathbb{Z}_q$ under the following constraints: for $\hat{i} \in [Q]$, $\sum_{k\in\Delta_{\hat{i}}} b_k = \lceil q/2 \rceil \mu$. Set $\mathbf{b}_k = [0, \ldots, 0, b_k]$.
     - Sample random vectors $\{\widetilde{\beta}_k\}_{k\in[N]} \in \mathbb{Z}_q^m$ condition on the following equations:
     $$\textstyle\sum_{k\in\Delta_{\hat{i}}} \widetilde{\beta}_k = \mathbf{K}_{\hat{i}}^\top \cdot \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}_{\hat{i}}} \end{pmatrix} \text{ for } \hat{i} \in [Q'].$$
     - Sample $\mathbf{e}'_k \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ for $k \in [N]$, Set $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}'_k + \mathbf{b}_k \mod q$.
   - It outputs the simulated ciphertext
   $$\mathsf{ct}^* := (\Psi, \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]}).$$
4. $\mathsf{QKeyGen}^*_{\mathsf{post}}(\mathsf{st}, f)$ generates as in the real $\mathsf{QPE.KeyGen}$ algorithm for all 0-key queries. Otherwise, assume that the current state contains $Q'(< Q)$ tuples of $(f_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{K}_{\hat{i}}))$ and corresponding functionality value, for a 1-key query $f_{\hat{i}_p}$, the simulator computes as follows.
   - If the adversary did not make 1-key queried before the challenge, i.e., $Q = 0$, then update $\mathsf{st} := \emptyset \cup \mu$. Next, set $\Delta = \Delta_{\hat{i}_p}$ if $\mu = 0$, otherwise as $\Delta = \Delta'_{\hat{i}_p}$.
   - If the current state is not empty, then set $\Delta = \Delta_{\hat{i}_p}$ for which is chosen during QEnc^* algorithm.
   - Compute $\mathbf{P}_\Delta = \sum_{k\in\Delta} \mathbf{P}_k$ and $\widetilde{\beta}_\Delta = \sum_{k\in\Delta} \widetilde{\beta}_k$, where $\{\widetilde{\beta}_k\}_{k\in[N]}$ are chosen during QEnc^*.
   - Compute $\Psi_f := \mathsf{HEval}_f(\Psi)$, $\mathbf{H}_{\hat{f}}$ and $\mathbf{H}_{\hat{f},\psi'}$, and use these results to compute $\mathbf{B}_{\hat{f}}$ and $\mathbf{c}_{\hat{f}}$, respectively.

– Sample $\mathbf{J}_{\hat{i}_p} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$, use $\mathbf{T}_{\mathbf{B}'}$ to sample $\begin{bmatrix}\mathbf{K}_{\hat{i}_p,1}\\ \mathbf{K}_{\hat{i}_p,2}\end{bmatrix}$ by $\mathsf{SampleLeft}$ such that

$$\begin{bmatrix}\mathbf{B} & \mathbf{B}_{f_{\hat{i}_p}}\\ \mathbf{z}^\top & \mathbf{c}_{f_{\hat{i}_p}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{K}_{\hat{i}_p,1}\\ \mathbf{K}_{\hat{i}_p,2}\end{bmatrix} = \begin{bmatrix}\mathbf{P}_{\Delta}\\ \widetilde{\beta}_{\Delta}^\top\end{bmatrix} - \begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix} \cdot \mathbf{J}_{\hat{i}_p}.$$

– Set $\mathbf{K}_{f_{\hat{i}_p}} = \begin{bmatrix}\mathbf{J}_{\hat{i}_p} + \mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$ and output $\mathsf{sk}_{f_{\hat{i}_p}} := (\Delta, \mathbf{K}_{f_{\hat{i}_p}})$.

Auxiliary Algorithms.

$\mathsf{QPE.Setup}_1^*(1^\lambda, \mathbf{x}^*)$: Do the following:

1. Generate $(\mathbf{B}, \mathbf{T}_\mathbf{B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
2. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, compute $\mathbf{z}^\top := \mathbf{s}^\top\mathbf{B} + \mathbf{e}^\top$.
3. Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ for $i \in [\ell]$ and compute $\Psi_i := \begin{pmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{pmatrix}\mathbf{R}_i + x_i^*\mathbf{G}$. Let $\psi = (\psi_1, \ldots, \psi_L)$ denote the bit-representation of $\Psi := [\Psi_1|\cdots|\Psi_\ell]$.
4. Set $\mathbf{B}_j = \mathbf{B}\cdot\mathbf{W}_j - \psi_j \cdot \overline{\mathbf{G}}$ for $j \in [L]$, where $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ for $j \in [L]$.
5. Choose Q random subset $(\Delta_1, \ldots, \Delta_Q)$ with size w according sampler $\mathsf{SamplerSet}(N, Q, w)$. By cover-freeness, for every $\hat{i} \in [Q]$, there exists a unique index $\delta_{\hat{i}}$ that only appears in $\Delta_{\hat{i}}$ but not the other subsets. Sample $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m}, \sqrt{\rho^2+s^2}}$ for $\hat{i} \in [Q]$.
6. Sample $\mathbf{P}_k \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ for $k \in [N]$ under the constraints that $\sum_{k\in\Delta_{\hat{i}}}\mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*$ for each $\hat{i} \in [Q]$. Denote $\sum_{k\in\Delta_{\hat{i}}}\mathbf{P}_k$ as $\mathbf{P}_{\Delta_{\hat{i}}}$.
7. Initialize $\mathsf{st} := \emptyset$. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \{\mathbf{P}_k\}_{k\in[N]}),$$
$$\mathsf{msk} := (\mathbf{T}_\mathbf{B}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{W}_j\}_{j\in[L]}, \{\mathbf{J}_{\hat{i}}^*\}_{\hat{i}\in[Q]}, \mathbf{s}).$$

$\mathsf{QPE.Enc}_1^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Do the following:

1. Set $\beta_0 := \mathbf{z}$.
2. For $j \in [L]$, compute $\mathbf{c}_j := \mathbf{W}_j^\top\beta_0$, where $\mathbf{W}_j$ are chosen in QSetup_1^*.
3. $\{\beta_{1,k}\}_{k\in[N]}$ is computed as real encryption algorithm, except that the secret randomness $\mathbf{s}$ is set the one chosen in QSetup_1^*.
4. Output the ciphertext $\mathsf{ct}^* := (\Psi, \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{QPE.KeyGen}_1^*(\mathsf{msk}, \mathsf{st}, f)$: This algorithm is stateful that keeps track of how many keys have been queried before. Particularly, it does the following:

1. Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute the homomorphic public key corresponding to circuit $\hat{f}$ as

$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}),$$

$$\begin{aligned}\mathbf{B}_{\hat{f}} &:= [\mathbf{B}_1 \mid \cdots \mid \mathbf{B}_L]\cdot\mathbf{H}_{\hat{f}}\\ &= [\mathbf{B}_1 + \psi_1\overline{\mathbf{G}} \mid \cdots \mid \mathbf{B}_L + \psi_L\overline{\mathbf{G}}]\cdot\mathbf{H}_{\hat{f},\psi} - \overline{\Psi}_f\\ &= \mathbf{B}[\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L]\cdot\mathbf{H}_{\hat{f},\psi} - \overline{\Psi}_f\\ &= \mathbf{B}(\mathbf{W}_{\hat{f}} - \mathbf{R}_f) - f(\mathbf{x}^*)\overline{\mathbf{G}}\end{aligned}$$

where $\mathbf{W}_{\hat{f}} := [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi}$, $\Psi_f = \begin{pmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{pmatrix} \mathbf{R}_f + f(\mathbf{x}^*) \begin{pmatrix} \overline{\mathbf{G}} \\ \underline{\mathbf{G}} \end{pmatrix}$.

2. For 0-key query such that $f(\mathbf{x}^*) \neq 0$, first sample a fresh random subset $\Delta \subseteq [N]$ with cardinality w according sampler $\mathsf{SamplerSet}(N, Q, w)$, then generate

$$\mathbf{K}_f \leftarrow \mathsf{SampleRight}(\mathbf{B}, \mathbf{G}, \mathbf{W}_{\hat{f}} - \mathbf{R}_f, \textstyle\sum_{k \in \Delta} \mathbf{P}_k, s),$$

satisfying $[\mathbf{B} | \mathbf{B}_{\hat{f}}] \cdot \mathbf{K}_f = \mathbf{P}_\Delta = \sum_{k \in \Delta} \mathbf{P}_k$.

3. For 1-key query $f = f_{\hat{i}}$ that $f_{\hat{i}}(\mathbf{x}^*) = 0$, the algorithm does the following. We use index $\hat{i} \in [Q]$ to denote the number of overall 1-key queries currently.
   - Set $\Delta = \Delta_{\hat{i}}$ instead of sampling it freshly. Recall that $\Delta_{\hat{i}}$ is sampled in the $\mathsf{QPE.Setup}_1^*$, so as $\mathbf{J}_{\hat{i}}^*$.
   - Sample $\mathbf{K}_{\hat{i},2} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m}, s}$, and set

$$\mathbf{K}_{f_{\hat{i}}} := \begin{bmatrix} \mathbf{J}_{\hat{i}}^* - (\mathbf{W}_{\hat{f}_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}}) \cdot \mathbf{K}_{\hat{i},2} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix}.$$

   Then, by the construction, we have

$$[\mathbf{B} | \mathbf{B}_{\hat{f}_{\hat{i}}}] \cdot \mathbf{K}_f = [\mathbf{B} | \mathbf{B}(\mathbf{W}_{\hat{f}_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}})] \cdot \mathbf{K}_f = \mathbf{B}\mathbf{J}_{\hat{i}}^* = \sum_{k \in \Delta} \mathbf{P}_k.$$

   - Update $\mathsf{st} := \mathsf{st} \cup (f_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{K}_{\hat{i}}))$.

4. Return $\mathsf{sk}_f = (\Delta, \mathbf{K}_f)$.

$\mathsf{QPE.Setup}_2^*(1^\lambda, \mathbf{x}^*)$: Do the following:

1. Sample $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m}, \sqrt{\rho^2 + s^2}}$ for $\hat{i} \in [Q]$.
2. Sample $\mathbf{J}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m}, \sqrt{\rho^2 + s^2}}$ for $k \in [N] \backslash \{\delta_1, \ldots, \delta_Q\}$, where $\delta_{\hat{i}}$ is a unique index that only appears in $\Delta_{\hat{i}}$ but not the other subsets. Set $\mathbf{J}_{\delta_{\hat{i}}} = \mathbf{J}_{\hat{i}}^* - \sum_{k \in \Delta_{\hat{i}}^* \backslash \{\delta_{\hat{i}}\}} \mathbf{J}_k$ for $\hat{i} \in [Q]$, then we have $\mathbf{J}_{\hat{i}}^* = \sum_{k \in \Delta_{\hat{i}}} \mathbf{J}_k$.
3. Sample $\mathbf{B}$ randomly and set $\mathbf{P}_k = \mathbf{B} \cdot \mathbf{J}_k$ for $k \in [N]$.
4. Compute and set the remaining components as in $\mathsf{QPE.Setup}_1^*$.

$\mathsf{QPE.Enc}_2^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Do the following:

1. Generate $\beta_0, \{\mathbf{c}_j\}_{j \in [L]}$ as in $\mathsf{QPE.Enc}_1^*$.
2. For $k \in [N]$, compute $\beta_{1,k}$ as follows:
   - Sample $\mathbf{e}'_k \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ and set $\mathbf{b}_k = [0, \ldots, 0, \frac{\lfloor q/2 \rceil}{w} \mu]^\top \in \mathbb{Z}_q^m$.
   - Compute $\beta_{1,k} = \mathbf{J}_k^\top \cdot \beta_0 + \mathbf{e}'_k + \mathbf{b}_k \mod q$.
3. Output $\mathsf{ct}^* := (\Psi, \beta_0, \{\beta_{1,k}\}_{k \in [N]}, \{\mathbf{c}_j\}_{j \in [L]})$.

$\mathsf{QPE.Setup}_3^*(1^\lambda, \mathbf{x}^*)$: Sample $\mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^m$. Compute and set the remaining components as in QSetup_2^*.
$\mathsf{QPE.KeyGen}_2^*(\mathsf{msk}, \mathsf{st}, f)$: Do the following:

1. For 0-key query, generate the key $\mathbf{K}_f$ as in $\mathsf{QPE.KeyGen}$.
2. For 1-key query, set the key $\mathbf{K}_f$ as in $\mathsf{QPE.KeyGen}_1^*$.
3. Output $\mathsf{sk}_f := (\Delta, \mathbf{K}_f)$.

$\mathsf{QPE.Setup}_4^*(1^\lambda, \mathbf{x}^*)$: Do the following:

1. Generate $(\mathbf{B}', \mathbf{T}_{\mathbf{B}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^m, q)$, then parse $\mathbf{B}' = \begin{bmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{bmatrix}$.

2. Define $\widetilde{\beta}_k = \mathbf{J}_k^\top \cdot \mathbf{z}$ for $k \in [N]$.
3. Set $\mathbf{B}$ as the public matrix in mpk.
4. Compute remaining elements as in $\mathsf{QPE.Setup}_2^*$. Additionally, add $\{\widetilde{\beta}_k\}_{k\in[N]}$ into msk.

$\mathsf{QPE.Enc}_3^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Compute and set the ciphertext components the same as in $\mathsf{QPE.Enc}_2^*$, except that $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}'_k + \mathbf{b}_k$ for $k \in [N]$.

$\mathsf{QPE.KeyGen}_3^*(\mathsf{msk}, \mathsf{st}, f)$: Do the following:

1. For 0-key query, generate the key $\mathbf{K}_f$ as in $\mathsf{QPE.KeyGen}_2^*$.
2. For 1-key query, let $f_{\hat{i}}$ be the $\hat{i}$-th 1-key query, set $\varDelta = \varDelta_{\hat{i}}$ ($\varDelta_{\hat{i}}$ is the subset sampled in the $\mathsf{QPE.Setup}_1^*$).
   - Sample $\mathbf{J}_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$. Use $\mathbf{T}_{\mathbf{B}'}$ to sample $\begin{bmatrix}\mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$ by $\mathsf{SampleLeft}$ such that
   $$\begin{bmatrix}\mathbf{B} & \mathbf{B}_{\hat{f}}\\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix} = -\begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix} \cdot \mathbf{J}_{\hat{i}} + \begin{bmatrix}\mathbf{P}_\varDelta\\ \widetilde{\beta}_\varDelta^\top\end{bmatrix}.$$
   - Set $\mathbf{K}_f = \begin{bmatrix}\mathbf{J}_{\hat{i}} + \mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$.
3. Output $\mathsf{sk}_f := (\varDelta, \mathbf{K}_f)$.

$\mathsf{QPE.Setup}_5^*(1^\lambda, \mathbf{x}^*)$: Do the following:

1. Sample $\mathbf{P}_k$ randomly from $\mathbb{Z}_q^{m\times n}$ under the constraint that $\sum_{k\in\varDelta_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*$, which is thus distributed exactly the same as in $\mathsf{QPE.Setup}_1^*$.
2. Sample $\widetilde{\beta}_k$ randomly from $\mathbb{Z}_q^m$ for under the constraint that $\sum_{k\in\varDelta_{\hat{i}}} \widetilde{\beta}_k = \mathbf{J}_{\hat{i}}^* \cdot \mathbf{z}$, denote $\sum_{k\in\varDelta_{\hat{i}}} \widetilde{\beta}_k$ as $\widetilde{\beta}_{\varDelta_{\hat{i}}}$.

$\mathsf{QPE.Setup}_6^*(1^\lambda, \mathbf{x}^*)$: Sample $\{\mathbf{B}_j\}$ and $\{\mathbf{P}_k\}$ as in the normal $\mathsf{QPE.Setup}$, and sample $\widetilde{\beta}_k \xleftarrow{\$} \mathbb{Z}_q^m$ for $k \in [N]$. The remaining components are generated as in $\mathsf{QPE.Setup}_5^*$.
$\mathsf{QPE.Enc}_4^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Sample $\{\mathbf{c}_j\}_{j\in[L]}$ and Ψ randomly. Compute the remaining components as in $\mathsf{QPE.Enc}_3^*$.
$\mathsf{QPE.Enc}_5^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mu)$: Generate random shares $\{b_k\}_{k\in[N]}$ over $\mathbb{Z}_q$ under the following constraints: for $\hat{i} \in [Q]$, $\sum_{k\in\varDelta_{\hat{i}}} b_k = \lceil q/2 \rceil \mu$. Compute the remaining components as in $\mathsf{QPE.Enc}_4^*$.

Hybrids.

$\mathcal{H}_0$: The real experiment.

$\mathcal{H}_1$: The real game algorithms $\mathsf{QPE.Setup}$ and $\mathsf{QPE.Enc}$ are replaced with $\mathsf{QPE.Setup}_1^*$ and $\mathsf{QPE.Enc}_1^*$, which use the knowledge of $\mathbf{x}^*$ to generate the public parameters, the master public/secret keys, and additionally samples random $\mathbf{P}_k$ under the constrain $\sum_{k\in\varDelta_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*$. $\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically close by an application of the Leftover Hash Lemma.

$\mathcal{H}_2$: The real game algorithm $\mathsf{QPE.KeyGen}$ is replaced with $\mathsf{QPE.KeyGen}_1^*$ where instead of using the trapdoor $\mathbf{T}_\mathbf{B}$ of the matrix $\mathbf{B}$, secret keys for a 0-key queries are sampled using the public trapdoor $\mathbf{T}_\mathbf{G}$ along with the trapdoor information generated in $\mathsf{QPE.Setup}_1^*$, and the secret key for the $\hat{i}$-th 1-key query for function $f_{\hat{i}}$ is generated as $\left(\varDelta_{\hat{i}}, \begin{bmatrix}\mathbf{J}_{\hat{i}}^* - (\mathbf{W}_{f_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}})\cdot\mathbf{K}_{\hat{i},2}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}\right)$.

$\mathcal{H}_3$: $\mathsf{QPE.Setup}_1^*$ is replaced by $\mathsf{QPE.Setup}_2^*$. In this hybrid, $\mathbf{B}$ is sampled randomly, the public matrices $\{\mathbf{P}_k\}$ are generated by first sampling matrices $\mathbf{J}_k$ from Gaussian distributions, then setting $\mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_k$.

$\mathcal{H}_4$: $\mathsf{QPE.Enc}_1^*$ is replaced by $\mathsf{QPE.Enc}_2^*$, in which $\beta_{1,k}$ is computed using β_0 and $\mathbf{J}_k$.

$\mathcal{H}_5$: $\mathsf{QPE.Setup}_2^*$ is replaced by $\mathsf{QPE.Setup}_3^*$, in which $\mathbf{z}$ is chosen from uniformly random and thus all ciphertext elements are derived from it.

$\mathcal{H}_6$: The algorithms $\mathsf{QPE.Setup}_3^*$ and $\mathsf{QPE.Enc}_2^*$ are replaced by $\mathsf{QPE.Setup}_4^*$ and $\mathsf{QPE.Enc}_3^*$, where the $\mathsf{TrapGen}$ algorithm outputs the public matrix $\mathbf{B}' = \begin{bmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{bmatrix}$ together with $\mathbf{T}_{\mathbf{B}'}$, the vector $\mathbf{z}$ is set as last row of output matrix $\mathbf{B}'$ from $\mathsf{TrapGen}$ algorithm instead of sampling uniformly and the vectors $\{\widetilde{\beta}_k\}_{k\in[N]}$ are added into the master secret key. $\mathsf{QPE.Enc}_3^*$ is almost the same as the $\mathsf{QPE.Enc}_2^*$ except that $\beta_{1,k}$ is computed using $\widetilde{\beta}_k$.

$\mathcal{H}_7$: $\mathsf{QPE.Setup}_4^*$ is replaced by $\mathsf{QPE.Setup}_5^*$, where $\mathbf{P}_k$ and $\widetilde{\beta}_k$ are instead sampled randomly under specific conditions.

$\mathcal{H}_8$: $\mathsf{QPE.KeyGen}_1^*$ is replaced by $\mathsf{QPE.KeyGen}_2^*$. The algorithm $\mathsf{QPE.KeyGen}_2^*$ is as the same as $\mathsf{QPE.KeyGen}_1^*$ except for the response to the 1-key queries.

$\mathcal{H}_9$: $\mathsf{QPE.KeyGen}_2^*$ is replaced by $\mathsf{QPE.KeyGen}_3^*$, where the responses to the 1-key queries are generated by using the trapdoor $\mathbf{T}_{\mathbf{B}'}$ such that

$$\begin{bmatrix} \mathbf{B} & \mathbf{B}_{\hat{f}} \\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{K}_{\hat{i},1} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_\Delta \\ \widetilde{\beta}_\Delta^\top \end{bmatrix} - \begin{bmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{bmatrix} \cdot \mathbf{J}_{\hat{i}}.$$

$\mathcal{H}_{10}$: The algorithms $\mathsf{QPE.Setup}_5^*$ and $\mathsf{QPE.Enc}_3^*$ are replaced by $\mathsf{QPE.Setup}_6^*$ and $\mathsf{QPE.Enc}_4^*$, where the public matrices $\{\mathbf{B}_j\}, \{\mathbf{P}_k\}$ are generated as the real world, and $\{\widetilde{\beta}_k\}_{k\in[N]}$ are sampled uniformly at random. Also the ciphertext components $\{\mathbf{c}_j\}_{j\in[L]}$ and Ψ are sampled uniformly.

$\mathcal{H}_{11}$: $\mathsf{QPE.Enc}_4^*$ is replaced by $\mathsf{QPE.Enc}_5^*$, where the secret sharing is generated following a different approach.

$\mathcal{H}_{12}$: The ideal experiment.

Next, we will prove that each pair of adjacent hybrid arguments is indistinguishable.

Lemma B.11 *$\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically indistinguishable.*

Proof. The difference between the two hybrids lies in how the public parameters and the ciphertext are generated.

1. The public parameters

   The matrix $\mathbf{B}$ in both two hybrids is generated using $\mathsf{TrapGen}$ algorithm. The difference of public parameters between the two hybrids is how the remaining public parameters are generated. $\{\mathbf{B}_j\}, \{\mathbf{P}_k\}$ are sampled from uniformly random in $\mathcal{H}_0$. In $\mathcal{H}_1$, $\mathbf{B}_j$ is set as $\mathbf{B}\mathbf{W}_j - \psi_j\overline{\mathbf{G}}$ for $\mathbf{W}_j \xleftarrow{\$} \{0,1\}^{m\times m}$. $\{\mathbf{P}_k\}_{k\in[N]}$ are sampled randomly under the condition $\sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*$ for $\hat{i}\in[Q]$, where $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$.

   Notice that $\mathbf{B}$ is generated using $\mathsf{TrapGen}$ algorithm, by Lemma 2.1, we have that $\mathbf{B} \overset{s}{\approx} \mathcal{U}$, where $\mathcal{U}$ denotes the uniform distribution over $\mathbb{Z}_q^{n\times m}$.

   By Leftover Hash Lemma (Lemma 2.6), we have $(\mathbf{B}, \mathbf{B}\mathbf{W}_j - \psi_j\overline{\mathbf{G}}, \mathbf{W}_j^\top\mathbf{e}) \overset{s}{\approx} (\mathbf{B}, \mathcal{U}, \mathbf{W}_j^\top\mathbf{e})$. Moreover, we have $(\mathbf{B}, \{\mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*\}_{\hat{i}\in[Q]}) \overset{s}{\approx} (\mathbf{B}, \{\mathcal{U}\}_{\hat{i}\in[Q]})$, by relying on Lemma 2.4. Hence, the matrices $\mathbf{P}_k$ is also distributed as uniform, as in $\mathcal{H}_0$.
2. The ciphertext

   The ciphertext components $\mathbf{c}_j$ are statistical close in $\mathcal{H}_0$ and $\mathcal{H}_1$, due to the similar reason in the proof of Lemma B.1.

□

□

Lemma B.12 *$\mathcal{H}_1$ and $\mathcal{H}_2$ are statistically indistinguishable.*

Proof. In both $\mathcal{H}_1$ and $\mathcal{H}_2$, due to the reason that in $\mathsf{QPE.Setup}_1^*$ the attribute encoding matrices $\mathbf{B}_j$ are programmed as $\mathbf{B}\mathbf{W}_j - \psi_j\overline{\mathbf{G}}$, the encoding matrix $\mathbf{B}_{\hat{f}}$ after homomorphic evaluation is in the exact same form $\mathbf{B}(\mathbf{W}_{\hat{f}} - \mathbf{R}_f) - f(\mathbf{x}^*)\overline{\mathbf{G}}$ as described in $\mathsf{QPE.KeyGen}_1^*$.

The difference between the two hybrids is in the way the queried secret keys are generated. We consider the following two cases:

1. For the 0-key query of f, in $\mathcal{H}_1$, these keys are sampled using the $\mathsf{SampleLeft}$ algorithm, whereas in $\mathcal{H}_2$, they are sampled using the $\mathsf{SampleRight}$ algorithm. By employing the lemma 2.3, the resulting distributions are statistically indistinguishable.
2. For the 1-key query of f, we note that $\{\mathbf{P}_k\}$ are chosen the same way in both $\mathcal{H}_1$ and $\mathcal{H}_2$.
   - In $\mathcal{H}_1$, for $\hat{i} \in [Q]$, we sample $\mathbf{K}_{f_{\hat{i}}} = \begin{bmatrix} \mathbf{J}_{\hat{i}}^* + \mathbf{K}_{\hat{i},1} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix}$ by firstly sampling $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, then using $\mathsf{SampleLeft}$ algorithem to sample $\begin{bmatrix} \mathbf{K}_{\hat{i},1} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix}$ such that
   $$[\mathbf{B} \mid \mathbf{B}_{\hat{f}}] \cdot \begin{bmatrix} \mathbf{K}_{\hat{i},1} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix} = \sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k - \mathbf{B}\mathbf{J}_{\hat{i}}^* \mod q,$$
   where the marginal distribution of $\sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\mathbf{J}_{\hat{i}}^*$ is uniformly at random, as $\mathbf{J}_{\hat{i}}^*$ is hidden in the view of adversary in this case.
   - In $\mathcal{H}_2$, for $\hat{i} \in [Q]$, we generate $\mathbf{K}_{f_{\hat{i}}} := \begin{bmatrix} \mathbf{J}_{\hat{i}}^* - (\mathbf{W}_{\hat{f}_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}}) \cdot \mathbf{K}_{\hat{i},2} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix}$, where $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$, $\mathbf{K}_{\hat{i},2} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$.

   Just as we previously analyzed the indistinguishability in proving Lemma B.2 for the (1,poly)-PE scheme, the key generation in $\mathcal{H}_1$ is exactly the procedure of $\mathsf{Sample\text{-}1}$, while in $\mathcal{H}_2$, these keys are sampled using the $\mathsf{Sample\text{-}2}$. Hence, by Theorem 2.1, the two cases are statistically indistinguishable. Via a simple hybrid argument, the indistinguishability also holds for Q key queries.

This completes the proof. □

□

Lemma B.13 *$\mathcal{H}_2$ and $\mathcal{H}_3$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way how the public matrices $\{\mathbf{P}_k\}$ are generated. In $\mathcal{H}_2$, $\mathbf{P}_k$ is chosen from uniformly random under the constrain that $\sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*$, where $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $\hat{i} \in [Q]$. In $\mathcal{H}_3$, $\mathbf{P}_k$ is set as $\mathbf{P}_k = \mathbf{B} \cdot \mathbf{J}_k$, where $\mathbf{J}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $k \in [N]\backslash\{\delta_1, \ldots, \delta_Q\}$ and $\mathbf{J}_{\delta_{\hat{i}}} = \mathbf{J}_{\hat{i}}^* - \sum_{k\in\Delta_{\hat{i}}^*\backslash\{\delta_{\hat{i}}\}} \mathbf{J}_k$ for $\hat{i} \in [Q]$. In other words, it still holds that $\mathbf{J}_{\hat{i}}^* = \sum_{k\in\Delta_{\hat{i}}} \mathbf{J}_k$. Hence, $\sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k = \sum_{k\in\Delta_{\hat{i}}} \mathbf{B} \cdot \mathbf{J}_k = \mathbf{B} \cdot \sum_{k\in\Delta_{\hat{i}}} \mathbf{J}_k = \mathbf{B} \cdot \mathbf{J}_{\hat{i}}^*$, as in $\mathcal{H}_2$. As $\{\mathbf{P}_k\}_{k\in[N]}$ in the two hybrids are under the same constraint, it remains to analyze the marginal distribution of $\mathbf{P}_k$ in $\mathcal{H}_3$. By lemma 2.6 and the parameter settings, we have $(\mathbf{B}, \{\mathbf{B} \cdot \mathbf{J}_k\}_{k\in[N]}) \overset{s}{\approx} (\mathbf{B}, \{\mathcal{U}\}_{k\in[N]})$. □

□

Lemma B.14 *$\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way how the ciphertext element $\{\beta_{1,k}\}_{k\in[N]}$ is generated. In details, we have

$$\begin{aligned}
\beta_{1,k} &= \mathbf{J}_k^\top \cdot \beta_0 + \mathbf{e}'_k + \mathbf{b}_k \text{ (in } \mathcal{H}_4) \\
&= \mathbf{B}\mathbf{J}_k^\top \cdot \mathbf{s} + \mathbf{J}_k^\top \cdot \mathbf{e} + \mathbf{e}'_k + \mathbf{b}_k \\
&= \mathbf{P}_k^\top \cdot \mathbf{s} + \mathbf{J}_k^\top \cdot \mathbf{e} + \mathbf{e}'_k + \mathbf{b}_k \\
&\overset{s}{\approx} \mathbf{P}_k^\top \cdot \mathbf{s} + \mathbf{e}'_k + \mathbf{b}_k \text{ (in } \mathcal{H}_3)
\end{aligned}$$

The last $\overset{s}{\approx}$ relies on noise flooding $\mathbf{e}'_k \overset{s}{\approx} \mathbf{J}_k^\top \cdot \mathbf{e} + \mathbf{e}'_k$ and our parameter choices. $\square$

$\square$

Lemma B.15 *$\mathcal{H}_4$ and $\mathcal{H}_5$ are computationally indistinguishable under the* LWE *assumption.*

The proof of this lemma is similar to the proof of Lemma B.4 and is therefore omitted for brevity.

Lemma B.16 *$\mathcal{H}_5$ and $\mathcal{H}_6$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is how the public matrix $\mathbf{B}$, the trapdoor and the vector $\mathbf{z}$ are generated. $\mathbf{B}$ and $\mathbf{z}$ are sampled randomly in $\mathcal{H}_6$. In $\mathcal{H}_7$, we first run $\mathsf{TrapGen}(1^{n+1}, 1^m, q)$ to get $(\mathbf{B}', \mathbf{T}_{\mathbf{B}'})$, then parse $\mathbf{B}' = \begin{bmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{bmatrix}$. By applying the property of $\mathsf{TrapGen}$ (Lemma 2.1), $(\mathbf{B}, \mathbf{z})$ in these two cases are both statistically close to the uniform distribution, and thus they are indistinguishable in two hybrids..

Furthermore, as $\widetilde{\beta}_k$ is set as $\mathbf{J}_k^\top \mathbf{z}$ and $\beta_0 \coloneqq \mathbf{z}$, $\beta_{1,k}$ is identically distributed in the two hybrids. $\square$

$\square$

Lemma B.17 *$\mathcal{H}_6$ and $\mathcal{H}_7$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is how the public matrices $\{\mathbf{P}_k\}$ and vectors $\{\widetilde{\beta}_k\}$ are generated.

Analogously to the proof of indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_3$ (Lemma B.13), the matrices $\{\mathbf{P}_k\}$ are statistically indistinguishable in two hybrids.

For $\widetilde{\beta}_k$, in $\mathcal{H}_7$, $\widetilde{\beta}_k = \mathbf{J}_k^\top \cdot \mathbf{z}'$, where $\mathbf{J}_k$ satisfies the constraint $\mathbf{J}^*_{\hat{i}} = \sum_{k \in \Delta_{\hat{i}}} \mathbf{J}_k$ for $\hat{i} \in [Q]$. By Lemma 2.6, the marginal distribution of $\{\widetilde{\beta}_k\}$ statistically close to uniformly random distribution under the constraint $\sum_{k \in \Delta_{\hat{i}}} \widetilde{\beta}_k = \mathbf{J}^*_{\hat{i}} \cdot \mathbf{z}$ for $k \in [N]$, which is exactly the distribution of $\{\widetilde{\beta}_k\}$ in $\mathcal{H}_8$. $\square$

$\square$

Lemma B.18 *$\mathcal{H}_7$ and $\mathcal{H}_8$ are statistically indistinguishable.*

Proof. The proof is analogous to the proof of indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ for generating secret keys of 0-key queries. Specifically, based on Lemma 2.1, $\mathbf{T}_{\mathbf{B}'}$ is also a trapdoor of $\mathbf{B}$ for Key generation algorithm. As the secret keys are sampled from the same Gaussian distribution over the same lattice for these two hybrids, it does not matter which trapdoor is used. $\square$

$\square$

Lemma B.19 *$\mathcal{H}_8$ and $\mathcal{H}_9$ are statistically indistinguishable.*

Proof. The only difference between the two hybrids lies in the way how the Q 1-keys are generated. In $\mathcal{H}_8$, the secret keys are sampled using the $\mathsf{QSampler}$-2 without trapdoor, while in $\mathcal{H}_9$, the secret keys are sampled using the $\mathsf{QSampler}$-1 with $\mathbf{T}_{\mathbf{B}'}$. Thus, by Lemma A.6, the two hybrids are statistically indistinguishable. $\square$

$\square$

Lemma B.20 *$\mathcal{H}_9$ and $\mathcal{H}_{10}$ are statistically indistinguishable.*

Proof. The only difference between the two hybrids lies in the way of how the public parameters and the ciphertext are generated.

– $\{\mathbf{B}_j\}_{j \in [L]}, \{\mathbf{P}_k\}_{k \in [N]}$
  The indistinguishability of the distribution of $\{\mathbf{B}_j\}, \{\mathbf{P}_k\}$ in $\mathcal{H}_9$ and $\mathcal{H}_{10}$ follows similarly as the proof of indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$.
– $\{\widetilde{\beta}_k\}_{k \in [N]}$
  In $\mathcal{H}_9$, the marginal distribution of $\{\widetilde{\beta}_k\}$ is statistically close to uniformly random according to Lemma 2.6, as in $\mathcal{H}_{10}$.

– $\{\mathbf{B}_j\}_{j\in[L]}$

In $\mathcal{H}_9$, the ciphertext elements are set as $\mathbf{c}_j = \mathbf{W}_j^\top \beta_0$. While, in $\mathcal{H}_{10}$, $\mathbf{c}_j$ are chosen independently and randomly from $\mathbb{Z}_q^m$.

Recall that $\beta_0 \overset{s}{\approx} \mathcal{U}$ in both hybrids, thus the indistinguishability of two hybrids follows from the Leftover Hash Lemma (Lemma 2.6):

$$(\mathbf{B}, \beta_0, \{\mathbf{B}\mathbf{W}_j, \mathbf{W}_j^\top \beta_0\}) \overset{s}{\approx} (\mathbf{B}, \beta_0, \{\mathcal{U}, \mathcal{U}\}).$$

– $\Psi = (\Psi_1, \ldots, \Psi_\ell)$

In $\mathcal{H}_9$, the FHE ciphertext elements Ψ_i are computed by

$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G},$$

where $\mathbf{R}_i$ is the randomness chosen independently from $\{0,1\}^{m\times m}$. From Leftover Hash Lemma (Lemma 2.6), we have that $\begin{pmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{pmatrix} \mathbf{R}_i$ is statistically close to uniform. Therefore, $\Psi_i \overset{s}{\approx} \mathcal{U}$, as in $\mathcal{H}_{10}$.

Hence, the two hybrids are statistically close. □

□

Lemma B.21 *$\mathcal{H}_{10}$ and $\mathcal{H}_{11}$ are statistically indistinguishable.*

Proof. The only difference between the two hybrids is the way to generate the message encoding vectors $\mathbf{b}_k$ in the ciphertext elements $\beta_{1,k}$ for $k \in [N]$. In both hybrids, $\beta_{1,k}$ is set as $\widetilde{\beta}_k + \mathbf{e}'_k + \mathbf{b}_k$ for $\widetilde{\beta}_k \xleftarrow{\$} \mathbb{Z}_q^m$. In addition, secret keys for 1-keys are generated to satisfy the constraint

$$\begin{bmatrix} \mathbf{B} & \mathbf{B}_{\hat{f}_{\hat{i}}} \\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}_{\hat{i}}} \end{bmatrix} \cdot \mathsf{sk}_{f_{\hat{i}}} = \begin{bmatrix} \mathbf{P}_{\Delta_{\hat{i}}} \\ \widetilde{\beta}^\top_{\Delta_{\hat{i}}} \end{bmatrix},$$

which will guarantee the correctness of the decryption. In other words, $\widetilde{\beta}_k$ plays the role of a one-time pad in $\beta_{1,k}$ to hide message pieces $\mathbf{b}_k$ if no 1-key has ever been queried. On the other hand, the adversary will only learn the value $\sum_{\Delta_{\hat{i}}} b_k = \lceil q/2 \rceil \mu$ given the secret key for 1-key query $\hat{f}_{\hat{i}}$. The value $\sum_{\Delta_{\hat{i}}} b_k$ is set to be identical in both hybrids. Therefore, $\mathcal{H}_{10}$ and $\mathcal{H}_{11}$ are statistically indistinguishable.

□

□

Lemma B.22 *$\mathcal{H}_{11}$ and $\mathcal{H}_{12}$ are statistically indistinguishable.*

Proof. From the viewpoint of the adversary, the available transcript after the whole experiment will contain $(\mathsf{mpk}, \{\mathsf{sk}_{f_i}\}_{i\in\mathsf{poly}}, \{\mathsf{sk}_{f_{\hat{i}}}\}_{\hat{i}\in[Q]}, \mathsf{ct}^*)$. We observe that the distribution of $(\mathsf{mpk}, \{\mathsf{sk}_{f_i}\}_{i\in\mathsf{poly}})$ are identical in both hybrids. It remains to show the indistinguishability of the remaining components in two hybrids.

The only difference for generating ct^* between the two hybrids is the procedure of message secret-sharing. As analyzed in Lemma B.21, the distributions of ct^* in two hybrids are statistically close as long as the summation $\sum_{\Delta_{\hat{i}}} b_k$ stays the same.

To show the statistical indistinguishability of $\{\mathsf{sk}_{f_{\hat{i}}} = (\Delta_{\hat{i}}, \mathbf{K}_{\hat{i}})\}_{\hat{i}\in[Q]}$, we firstly consider the distribution of $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ in the following two cases:

1. If there is no pre-challenge 1-key query, then all Q subsets are chosen during $\mathsf{QPE.Setup}_6^*$ phase in $\mathcal{H}_{11}$, while these subsets are all sampled in $\mathsf{QPE.Enc}^*$ in $\mathcal{H}_{12}$. The distributions of Q subsets in the two hybrids are thus identical.

2. If the adversary has queried Q' 1-keys in the pre-challenge phase, then in $\mathcal{H}_{12}$, each of the first Q' subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q']}$ is sampled independently during each 1-key generation in the pre-challenge key query phase, while the remaining $Q-Q'$ subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q'+1,Q]}$ are sampled in $\mathsf{QPE.Enc}^*$. In $\mathcal{H}_{11}$, all the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ are chosen during $\mathsf{QPE.Setup}_6^*$. As each subset $\Delta_{\hat{i}}$ is sampled independently, the distribution of $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ in the two hybrids are identical according to Lemma 2.7.

In addition, the distributions of $\mathbf{K}_{\hat{i}}$ in the two hybrids are statistically close, since the generating approaches are identical. Therefore, $\mathcal{H}_{11}$ and $\mathcal{H}_{12}$ are statistically indistinguishable. □

□

This completes the security proof.

□

□

C Supplementary Material of Section 4

In this section, we provide the parameters setting and security proofs for the predicate encryption schemes described in Section 4, which were omitted from the main text due to space limitations.

C.1 Supplementary Material of (1, poly) P-IPFE Scheme in Section 4.1

Parameter Setting. We choose the parameters so that correctness and security of the scheme are satisfied. We must satisfy the following constraints.

1. For the security and correctness of ALS scheme, we set n_{ALS}, m_{ALS}, q_{ALS}, σ_{ALS}, ρ_{ALS}, α_{ALS} as chosen in the IPFE scheme described in Appendix D.1.
2. For correctness, the final magnitude of error obtained must be below $q/2Y$.
3. We set s used in $\mathsf{SampleLeft}$ (Lemma 2.2) and $\mathsf{SampleRight}$ (Lemma 2.3) such that the output matrices are statistically indistinguishable.
4. We must choose m large enough for the algorithm $\mathsf{TrapGen}$ (Lemma 2.1).
5. We must choose s_B such that LWE_{q,n,s_B} assumption holds.
6. We set s and ρ to meet the requirements of two-stage sampling techniques (Theorem 2.1).
7. We must choose the parameter s_D used to sample the error $\mathbf{e}_1$ in β_1 large enough so that the following relationship is satisfied:
$$\mathbf{e}_1 \overset{s}{\approx} \mathbf{J}^{*\top} \cdot \mathbf{e}_0 + \mathbf{e}_1 \text{ for } \mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times t},\sqrt{\rho^2+s^2}}.$$
8. We require $\tau > s_1(\mathbf{J}^*)$ in order to rely on ReRand algorithm for security proof. According to Lemma A.1, $s_1(\mathbf{J}^*)$ is bounded by $1/\sqrt{2\pi}\cdot\sqrt{\rho^2+s^2}\cdot(\sqrt{t}+\sqrt{m}+\sqrt{\lambda})$.

Our parameters may be chosen as: $n=\mathsf{poly}(\lambda)$, $m=(n+1)\log q$, $s_B=\omega(\sqrt{\log n})$, $s=O(Ln\log q)^{O(\hat{d})}\cdot\omega(\sqrt{\log m})$, $\rho=O(Ln\log q)^{O(\hat{d})}\cdot\omega(\sqrt{\log m})\cdot\lambda^{\omega(1)}$, $\tau=\sqrt{\rho^2+s^2}\cdot(\sqrt{t}+\sqrt{m}+\sqrt{\lambda})$, $s_D=\rho\cdot m\cdot\omega(\sqrt{\log n})\cdot\lambda^{\omega(1)}$, $q=2s_D\cdot t\cdot V\cdot Y$, where $L=\ell\cdot(n+1)^2\log q^2$, $\hat{d}=d\cdot O(\log m\log\log q)$.

Security.

Theorem (Restatement of Theorem 4.1) *Assuming the hardness of* LWE*, then the scheme described in Section 4.1 is a* $\mathsf{P\text{-}IPFE}$ *for the predicate class* $\mathcal{F}$*, message vector space* $\mathcal{U}$ *and key vector space* $\mathcal{V}$*, achieving* $(1,\mathsf{poly})$-sel-*SIM security that allows at most single 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

Proof. We define a PPT simulator Sim and prove that for any PPT adversary $\mathcal{A}$, the ideal experiment with respect to Sim is computationally indistinguishable (under the LWE assumption) from the output of the real experiment.

Simulator. $\mathsf{Sim}(1^\lambda, 1^{|\mathbf{x}|}, 1^{|\mathbf{u}|})$:

1. $\mathsf{Setup}^*(1^\lambda, 1^{|\mathbf{x}|}, 1^{|\mathbf{u}|})$ generates all public parameters as in the real Setup and initializes $\mathsf{st} := \emptyset$.
2. $\mathsf{KeyGen}^*_{\mathsf{pre}}(\mathsf{st}, f, \mathbf{v})$: It generates all secret keys as in the real KeyGen.
3. $\mathsf{Enc}^*(\mathsf{st})$: It takes as input st that contains $d^* = \langle \mathbf{u}^*, \mathbf{v}\rangle$ if the adversary has queried for $(f, \mathbf{v})$ such that $f(\mathbf{x}^*) = 0$ before the challenge query, then constructs the challenge ciphertext as follows.
   (a) It samples $\beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ independently and uniformly from $\mathbb{Z}_q^m$.
   (b) Samples $\{\Psi_i, \Psi'_i\}_{i\in[\ell]}$ uniformly from $\mathbb{Z}_q^{(n+1)\times(n+1)\log q}$.
   (c) If $\mathsf{st} = \emptyset$, it randomly samples β_1 from $\mathbb{Z}_q^m$.
   (d) If $\mathsf{st} = (f, \mathbf{v}, \mathsf{sk}_{f,\mathbf{v}}, d^* = \langle \mathbf{u}^*, \mathbf{v}\rangle)$, then Enc^* generates β_1 to satisfy the decryption consistency as follows.
      – Let $\Psi_f := \mathsf{HEval}_f(\Psi)$, let $\hat{f}$ denote the circuit computing $\Psi' \mapsto \overline{\Psi}'_f$, compute $\mathbf{H}_{\hat{f},\psi'} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}, \psi')$.
      – Set $\mathbf{c}_{\hat{f}}^\top := [\mathbf{c}_1^\top | \dots | \mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f$.
      – Compute $\beta_1 = \mathbf{K}_f^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}} \end{pmatrix} + \mathbf{e}_1 + \left\lfloor \frac{q}{Y} \right\rfloor \widetilde{\mathbf{u}}$, for $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^t, s_D}$ and $\widetilde{\mathbf{u}}$ such that $\langle \widetilde{\mathbf{u}}, \mathbf{v}\rangle = d^*$.
   (e) It outputs the simulated ciphertext
   $$\mathsf{ct}^* := (\Psi, \Psi', \beta_0, \beta_1, \{\mathbf{c}_j\}_{j\in[L]}).$$
4. $\mathsf{KeyGen}^*_{\mathsf{post}}(\mathsf{st}, f, \mathbf{v})$: It generates all secret keys as in the real KeyGen.

Auxiliary Algorithms.

$\mathsf{Setup}_1^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Do the following:

1. Sample $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
2. Sample $\mathbf{s}' \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}'_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, compute $\mathbf{z}'^\top = \mathbf{s}'^\top \mathbf{B} + \mathbf{e}'^\top_0$.
3. Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ for $i \in [\ell]$ and compute
   $$\Psi'_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{z}'^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G}.$$
   Let $\psi'_1, \dots, \psi'_L$ denote the bit-representation of $\Psi := [\Psi'_1 | \cdots | \Psi'_\ell]$.
4. Let $\mathbf{B}_j = \mathbf{B} \cdot \mathbf{W}_j - \psi'_j \cdot \overline{\mathbf{G}}$ for $j \in [L]$, where $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ for $j \in [L]$.
5. Sample $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times t}, \sqrt{\rho^2+s^2}}$ and set $\mathbf{P} = \mathbf{B} \cdot \mathbf{J}^* \mod q$.
6. Output the public and master secret keys.
   $$\mathsf{mpk} := (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P}),\ \mathsf{msk} := (\mathbf{T_B}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{W}_j\}_{j\in[L]}, \mathbf{J}^*).$$

$\mathsf{Enc}_1^*(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}^*, \mathbf{u}^*)$: Do the following:

1. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$ and $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^t, s_D}$.
2. Compute $\beta_0 := \mathbf{z} = \mathbf{B}^\top \mathbf{s} + \mathbf{e}_0$, $\beta_1 := \mathbf{P}^\top \mathbf{s} + \mathbf{e}_1 + \left\lfloor \frac{q}{Y} \right\rfloor \cdot \mathbf{u}^*$.
3. Compute $\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G}$ for $i \in [\ell]$, where $\mathbf{R}_i$ are chosen during Setup_1^*.
4. Compute $\mathbf{c}_j := \mathbf{W}_j^\top \beta_0$ for $j \in [L]$, where $\mathbf{W}_j$ re chosen during Setup_1^*.
5. Output the ciphertext $\mathsf{ct}^* := (\Psi, \Psi', \beta_0, \beta_1, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{KeyGen}_1^*(\mathsf{msk}, f, \mathbf{v})$: Do the following:

1. Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute

$$\mathbf{H}_{\hat{f}} := \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}),$$

$$\begin{aligned}\mathbf{B}_{\hat{f}} &:= [\mathbf{B}_1 \mid \cdots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}} \\ &= [\mathbf{B}_1 + \psi_1'\overline{\mathbf{G}} \mid \cdots \mid \mathbf{B}_L + \psi_L'\overline{\mathbf{G}}] \cdot \mathbf{H}_{\hat{f},\psi'} - \overline{\Psi}_f' \\ &= \mathbf{B}[\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi'} - \overline{\Psi}_f' \\ &= \mathbf{B}(\mathbf{W}_{\hat{f}} - \mathbf{R}_f) - f(\mathbf{x}^*)\overline{\mathbf{G}}\end{aligned}$$

   where

$$\mathbf{W}_{\hat{f}} := [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi'},\;\; \Psi_f' = \begin{pmatrix}\mathbf{B} \\ \mathbf{z}'^{\top}\end{pmatrix}\mathbf{R}_f + f(\mathbf{x}^*)\begin{pmatrix}\overline{\mathbf{G}} \\ \underline{\mathbf{G}}\end{pmatrix}.$$

2. For 0-key query such that $f(\mathbf{x}^*) \neq 0$, generate

$$\mathbf{K}_f \leftarrow \mathsf{SampleRight}(\mathbf{B}, \mathbf{G}, \mathbf{W}_{\hat{f}} - \mathbf{R}_f, \mathbf{P}, s).$$

3. For 1-key query such that $f(\mathbf{x}^*) = 0$, sample $\mathbf{K}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$, and set

$$\mathbf{K}_f := \begin{bmatrix}\mathbf{J}^* - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f) \cdot \mathbf{K}_2 \\ \mathbf{K}_2\end{bmatrix}.$$

4. Output $\mathsf{sk}_{f,\mathbf{v}} := \mathbf{K}_f \cdot \mathbf{v}$. Update $\mathsf{st} := \mathsf{st} \cup (f, \mathbf{v}, \mathsf{sk}_{f,\mathbf{v}}, d^* = \langle \mathbf{u}^*, \mathbf{v}\rangle)$ if there is a pre-challenge 1-key query $(f, \mathbf{v})$.

$\mathsf{Setup}_2^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Same as Setup_1^*, except that $\mathbf{B}$ is sampled uniformly from $\mathbb{Z}_q^{n\times m}$.

$\mathsf{Enc}_2^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$: Do the following:

1. Generate $\Psi, \Psi', \beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ as in Enc_1^*.
2. Sample $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}$, compute β_1 as follows:
   - If there is no pre-challenge 1-key query, then it computes β_1 as in Enc_1^*.
   - If the adversary has already queried the 1-key for $(f, \mathbf{v})$, then it first computes a vector $\widetilde{\mathbf{u}}$ satisfying $\langle \widetilde{\mathbf{u}}, \mathbf{v}\rangle = d^*$ and computes $\beta_1 = \mathbf{P}^{\top}\mathbf{s} + \mathbf{e}_1 + \left\lfloor \frac{q}{Y} \right\rfloor \widetilde{\mathbf{u}}$.
3. Output the ciphertext $\mathsf{ct}^* := (\Psi, \Psi', \beta_0, \beta_1, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{Enc}_3^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$: Do the following:

1. Generate $\Psi, \Psi', \beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ as in Enc_2^*.
2. Sample $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}$ and compute β_1 as follows:
   - If there is no pre-challenge 1-key query, then it computes $\beta_1 = \mathbf{J}^{*\top} \cdot \beta_0 + \mathbf{e}_1 + \mathbf{u}^*$.
   - If the adversary has already queried the 1-key for (f, v), then it computes $\beta_1 = \mathbf{K}_f^{\top} \cdot \begin{pmatrix}\beta_0 \\ \mathbf{c}_{\hat{f}}\end{pmatrix} + \mathbf{e}_1 + \widetilde{\mathbf{u}}$, where $\mathbf{c}_{\hat{f}}^{\top} := [\mathbf{c}_1^{\top} | \dots | \mathbf{c}_L^{\top}] \cdot \mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f$.
3. Output the ciphertext $\mathsf{ct}^* := (\Psi, \beta_0, \beta_1, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{Setup}_3^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Sample $\mathbf{z}' \xleftarrow{\$} \mathbb{Z}_q^m$. Compute and set the remaining components as in Setup_2^*.

$\mathsf{Enc}_4^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$: Sample $\mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^m$. Compute and set the remaining components as in Enc_3^*.

$\mathsf{Setup}_4^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Same as Setup_3^*, except that $\mathbf{B}$ is generated by running $\mathsf{TrapGen}$ algorithm.
$\mathsf{Enc}_5^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$: Sample $\mathbf{c}_j$ uniformly. If there is no pre-challenge 1-key, it samples β_1 randomly from $\mathbb{Z}_q^m$. Otherwise, compute β_1 as in Enc_4^*.
$\mathsf{Enc}_6^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st})$: Same as Enc_5^*, except that it additionally samples $\mathbf{z}, \mathbf{R}_i$ uniformly from corresponding distributions and computes Ψ' as in Enc_5^*.

Hybrids.

$\mathcal{H}_0$: The real experiment.
$\mathcal{H}_1$: The real game algorithms Setup and Enc are replaced with Setup_1^* and Enc_1^*. The challenge attribute $\mathbf{x}^*$ is used to generate the master public key and master secret key. Additionally, $\mathbf{P}$ is set as $\mathbf{B} \cdot \mathbf{J}^*$. By applying the Leftover Hash Lemma, $\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically close.
$\mathcal{H}_2$: The real game algorithms KeyGen is replaced with KeyGen_1^*, where, instead of using the trapdoor $\mathbf{T_B}$ of matrix $\mathbf{B}$, the secret keys for a 0-key queries are sampled using the public trapdoor $\mathbf{T_G}$, along with the trapdoor information generated in Setup_1^*, and the secret key for the 1-key query for the function $(f, \mathbf{v})$ is computed as $\mathbf{K}_f \coloneqq \begin{bmatrix} \mathbf{J} - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f) \cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix}$.
$\mathcal{H}_3$: Setup_1^* is replaced by Setup_2^*, where $\mathbf{B}$ is directly sampled uniformly from $\mathbb{Z}_q^{n \times m}$ instead of being generated by $\mathsf{TrapGen}$ algorithm.
$\mathcal{H}_4$: Enc_1^* is replaced by Enc_2^*, where β_1 is computed using $\widetilde{\mathbf{u}}$ instead of $\mathbf{u}^*$ if there is no 1-key queried before.
$\mathcal{H}_5$: Enc_2^* is replaced by Enc_3^*, where β_1 is computed from $\mathbf{J}^*$ if there is no 1-key queried before, or otherwise from the 1-key computed by KeyGen_1^*.
$\mathcal{H}_6$: Setup_2^* is replaced by Setup_3^*, where $\mathbf{z}'$ is chosen from uniformly random and public matrices $\{\mathbf{B}_j\}$ are derived from it.
$\mathcal{H}_7$: Enc_3^* is replaced by Enc_4^*, where $\mathbf{z}$ is chosen from uniformly random and thus the ciphertext elements $(\beta_0, \beta_1, \Psi', \{\mathbf{c}_j\}_{j \in [L]})$ are derived from it.
$\mathcal{H}_8$: Setup_3^* and KeyGen_1^* are replaced by Setup_4^* and KeyGen^*, respectively. Specifically, $\mathbf{B}$ is generated using $\mathsf{TrapGen}$ algorithm and all secret keys are computed by $\mathsf{SampleLeft}$ using $\mathbf{T_B}$.
$\mathcal{H}_9$: Enc_4^* is replaced by Enc_5^*, where $\{\mathbf{c}_j\}_{j \in [L]}$ and β_1 (in the case that there is no pre-challenge 1-key query) are chosen from uniformly random.
$\mathcal{H}_{10}$: Setup_4^* is replaced by Setup^*, where all public matrices $\{\mathbf{B}_j\}, \mathbf{P}$ are sampled uniformly without relying on the information of the challenge attribute $\mathbf{x}^*$. Additionally, Enc_5^* is switched to Enc_6^* to handle the sampling of $\mathbf{z}, \mathbf{R}_i, \mathbf{W}_j$, along with the computation of Ψ', ensuring consistency with with $\mathcal{H}_9$.
$\mathcal{H}_{11}$: Enc_6^* is replaced by Enc^*, where the FHE encryption (Ψ, Ψ') of $\mathbf{x}^*$ are directly chosen from uniformly random. Note that this hybrid is identical to the ideal experiment, specifically, no direct information about the challenge $(\mathbf{x}^*, \mathbf{u}^*)$ is provided to the simulator, except for the inner-product value that the adversary may obtain if a 1-key query is made prior to the challenge.

Next, we will prove that each pair of adjacent hybrid arguments is indistinguishable.

Lemma C.1 *$\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.1 and is therefore omitted for brevity.

Lemma C.2 *$\mathcal{H}_1$ and $\mathcal{H}_2$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.2 and is therefore omitted for brevity.

Lemma C.3 *$\mathcal{H}_2$ and $\mathcal{H}_3$ are statistically indistinguishable.*

Proof. The only difference between the two hybrids lies in how the public matrix $\mathbf{B}$ is generated. In $\mathcal{H}_2$, $\mathbf{B}$ is generated using $\mathsf{TrapGen}$ algorithm, and is therefore distributed statistically close to uniform, as it is when sampled directly in $\mathcal{H}_3$. □

□

Lemma C.4 *$\mathcal{H}_3$ and $\mathcal{H}_4$ are computationally indistinguishable assuming the security of ALS* IPFE *scheme.*

Proof. The difference between $\mathcal{H}_3$ and $\mathcal{H}_4$ lies in how the ciphertext component β_1 is computed when the adversary has queried for 1-key before the challenge query. We reduce the distinguishing advantage of $\mathcal{H}_4$ and $\mathcal{H}_3$ to the security of ALS IPFE scheme.

On receiving the public key $(\mathbf{A}_{\text{ALS}}, \mathbf{D}_{\text{ALS}}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{t\times n}$ from the ALS challenger (as described in Appendix D.1), we simulate the view of the distinguisher for $\mathcal{H}_3$ versus $\mathcal{H}_4$ as follows.

- Setup: Set $\mathbf{B} := \mathbf{A}_{\text{ALS}}^\top$. Sample $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times t}, \sqrt{\rho^2+s^2}}$ and set $\mathbf{P} := \mathbf{D}_{\text{ALS}}^\top + \mathbf{B}\mathbf{J}^*$. Compute $\{\mathbf{B}_j\}_{j\in[L]}$ as in Setup_2^*. Return the $\mathsf{mpk} = (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \mathbf{P})$ to the distinguisher.
- KeyGen: For key query $(f, \mathbf{v})$,
  - For the case where $f(\mathbf{x}^*) \neq 0$, generate the secret keys as in $\mathcal{H}_3$.
  - For the case where $f(\mathbf{x}^*) = 0$, set $\mathbf{v}' := \mathbf{v}$ and submit the key query $\mathbf{v}'$ to the ALS challenger, receiving $\mathsf{isk}_{\mathbf{v}'}$ in response. Then, compute and return the $\mathsf{sk}_{f,\mathbf{v}}$ as $\begin{bmatrix} \mathsf{isk}_{\mathbf{v}'} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{J}^* - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f)\cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix} \cdot \mathbf{v}$.
- Enc: Set $\mathbf{u}_0' = \mathbf{u}^*$. If no pre-challenge key query has been made, set $\mathbf{u}_1' = \mathbf{u}^*$. Otherwise, compute $\widetilde{\mathbf{u}}$ such that $\langle \widetilde{\mathbf{u}}, \mathbf{v}\rangle = \langle \mathbf{u}^*, \mathbf{v}\rangle$ for the 1-key query $(f, \mathbf{v})$ and set $\mathbf{u}_1' = \widetilde{\mathbf{u}}$. Send $(\mathbf{u}_0', \mathbf{u}_1')$ to the ALS challenger. Upon receiving $(\mathsf{ict}_0, \mathsf{ict}_1)$, compute as follows:

$$\beta_0 := \mathsf{ict}_0,\ \mathbf{c}_j^* := (\mathbf{W}_j^*)^\top \beta_0,$$
$$\beta_1 := \mathsf{ict}_1 + \mathsf{ReRand}(\mathbf{J}^*, \mathsf{ct}_0, \sigma_{\text{ALS}}, \tau) + \mathbf{e}_1 \text{ for } \mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^t, s_D},$$
$$\Psi_i := \begin{pmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G} \text{ for } i \in [\ell], \text{ where } \mathbf{z} := \beta_0.$$

  Return $\mathsf{ct}^* := (\Psi, \beta_0, \beta_1, \{\mathbf{c}_j\}_{j\in[L]})$.

Notice that all the queries submitted to the ALS challenger are admissible, as ensured by the setting of challenge vectors. For $\mathsf{ict}_0 = \mathbf{A}_{\text{ALS}} \cdot \mathbf{s} + \mathbf{e}_{\text{ALS},0}$, we know that $\mathsf{ReRand}(\mathbf{J}^*, \mathsf{ict}_0, \sigma_{\text{ALS}}, \tau) = (\mathbf{B}\mathbf{J}^*)^\top \cdot \mathbf{s} + \mathbf{e}'$ for $\tau > s_1(\mathbf{J}^*)$, where $\mathbf{e}' \overset{s}{\approx} \mathcal{D}_{\mathbb{Z}^t, 2\sigma_{\text{ALS}}\tau}$ by the property of ReRand (Lemma D.1). Therefore, we obtain the following result:

$$\begin{aligned}
\beta_1 &= \mathbf{D}_{\text{ALS}} \cdot \mathbf{s} + \mathbf{e}_{\text{ALS},1} + \left\lfloor \frac{q}{Y} \right\rfloor \cdot \mathbf{u}_b' + (\mathbf{B}\mathbf{J}^*)^\top \cdot \mathbf{s} + \mathbf{e}' + \mathbf{e}_1 \\
&= \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{e}_{\text{ALS},1} + \mathbf{e}' + \mathbf{e}_1 + \left\lfloor \frac{q}{Y} \right\rfloor \cdot \mathbf{u}_b' \\
&\overset{s}{\approx} \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{e}_1 + \left\lfloor \frac{q}{Y} \right\rfloor \cdot \mathbf{u}_b'.
\end{aligned}$$

The last $\overset{s}{\approx}$ relies on the noise flooding with suitable parameter choices. Hence, we successfully simulate the hybird $\mathcal{H}_3$ or $\mathcal{H}_4$ depending on the challenge bit chosen by the ALS challenger. □

□

Lemma C.5 *$\mathcal{H}_4$ and $\mathcal{H}_5$ are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way how the ciphertext element β_1 is generated.

1. If there is no pre-challenge 1-key query, then

$$\begin{aligned}\beta_1 &= \mathbf{J}^{*\top}\cdot\beta_0 + \mathbf{e}_1 + \mathbf{u}^* \text{ (in } \mathcal{H}_5)\\ &= \mathbf{B}\mathbf{J}^{*\top}\cdot\mathbf{s} + \mathbf{J}^{*\top}\cdot\mathbf{e}_0 + \mathbf{e}_1 + \mathbf{u}^*\\ &= \mathbf{P}^\top\cdot\mathbf{s} + \mathbf{J}^{*\top}\cdot\mathbf{e}_0 + \mathbf{e}_1 + \mathbf{u}^*\\ &\overset{s}{\approx} \mathbf{P}^\top\cdot\mathbf{s} + \mathbf{e}_1 + \mathbf{u}^* \text{ (in } \mathcal{H}_4)\end{aligned}$$

   In other words, β_1 in $\mathcal{H}_4$ and $\mathcal{H}_5$ are statistically close, as ensured by noise flooding (Lemma A.3) and our parameters setting .
2. If the adversary has already queried a 1-key for $(f, \mathbf{v})$, then

$$\beta_1 = \mathbf{K}_f^\top \cdot \begin{pmatrix}\beta_0\\ \mathbf{c}_{\hat{f}}\end{pmatrix} + \mathbf{e}_1 + \widetilde{\mathbf{u}} \text{ (in } \mathcal{H}_5)$$

   Notice that

$$\begin{aligned}\mathbf{c}_{\hat{f}}^\top &\coloneqq [\mathbf{c}_1^\top|\dots|\mathbf{c}_L^\top]\cdot\mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f\\ &= \beta_0^\top[\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L]\cdot\mathbf{H}_{\hat{f},\psi'} - \underline{\Psi}_f\\ &= \beta_0^\top\mathbf{W}_{\hat{f}} - \mathbf{z}^\top\mathbf{R}_f\\ &= \beta_0^\top(\mathbf{W}_{\hat{f}} - \mathbf{R}_f).\end{aligned}$$

   Hence, we have

$$\begin{aligned}\beta_1 &= \begin{bmatrix}\mathbf{J}^* - (\mathbf{W}_{\hat{f}} - \mathbf{R}_f)\cdot\mathbf{K}_2\\ \mathbf{K}_2\end{bmatrix}^\top \begin{pmatrix}\beta_0\\ (\mathbf{W}_{\hat{f}} - \mathbf{R}_f)^\top\beta_0\end{pmatrix} + \mathbf{e}_1 + \widetilde{\mathbf{u}} \text{ (in } \mathcal{H}_5)\\ &= \mathbf{J}^{*\top}\beta_0 + \mathbf{e}_1 + \widetilde{\mathbf{u}}\\ &\overset{s}{\approx} \mathbf{P}^\top\cdot\mathbf{s} + \mathbf{e}_1 + \widetilde{\mathbf{u}} \text{ (in } \mathcal{H}_4)\end{aligned}$$

   Similar to the first case, the resulting β_1 in $\mathcal{H}_5$ is statistically close to that in $\mathcal{H}_4$ due to noise flooding.

□

□

Lemma C.6 *$\mathcal{H}_5$ and $\mathcal{H}_6$ are computationally indistinguishable under the* LWE *assumption.*

Proof. We show how the LWE assumption can be broken given an adversary that distinguishes between $\mathcal{H}_4$ and $\mathcal{H}_5$. Given the LWE challenge sample $(\mathbf{B}, \mathbf{z}')$ where $\mathbf{z}'$ is either pseduorandom or truly random, run Setup_2^*, KeyGen_1^* and Enc_3^* accordingly. Note that if $\mathbf{z}' = \mathbf{B}^\top\mathbf{s} + \mathbf{e}_0'$, then we simulate the transcript of $\mathcal{H}_4$, otherwise that of $\mathcal{H}_5$ if $\mathbf{z}'$ is random. □

□

Lemma C.7 *$\mathcal{H}_6$ and $\mathcal{H}_7$ are computationally indistinguishable under the* LWE *assumption.*

The proof of this lemma is similar to the proof of Lemma C.6 and is therefore omitted for brevity.

Lemma C.8 *$\mathcal{H}_7$ and $\mathcal{H}_8$ are are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma C.2 and C.3, and is therefore omitted for brevity.

Lemma C.9 *$\mathcal{H}_8$ and $\mathcal{H}_9$ are are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.6 and B.9, and is therefore omitted for brevity.

Lemma C.10 *$\mathcal{H}_9$ and $\mathcal{H}_{10}$ are are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma C.1 and is therefore omitted for brevity.

Lemma C.11 *$\mathcal{H}_{10}$ and $\mathcal{H}_{11}$ are are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.10 and is therefore omitted for brevity.

□

□

C.2 Supplementary Material of (Q, poly) P-IPFE Scheme in Section 4.2

Parameter Setting. We choose the parameters so that correctness and security of the scheme are satisfied. We must satisfy the following constraints.

1. For the security and correctness of N-ALS scheme, we set n_{ALS}, m_{ALS}, q_{ALS}, σ_{ALS}, ρ_{ALS}, α_{ALS} as chosen in the N-ALS scheme as described in Appendix D.2.
2. For correctness, the final magnitude of error obtained must be below $p^{e-1}/2$.
3. We set s used in $\mathsf{SampleLeft}$ (Lemma 2.2) and $\mathsf{SampleRight}$ (Lemma 2.3) such that the output matrices are statistically indistinguishable.
4. We must choose m large enough for the algorithm $\mathsf{TrapGen}$ (Lemma 2.1).
5. We must choose s_B such that LWE_{q,n,s_B} assumption holds.
6. We set s and ρ to meet the requirements of two-stage sampling techniques (Theorem 2.1).
7. We must choose the parameter s_D used to sample the error $\mathbf{e}_{1,k}$ in β_k large enough so that the following equations are satisfied for $k \in [N]$:
$$\mathbf{e}_{1,k} \overset{s}{\approx} \mathbf{J}_k^\top \cdot \mathbf{e}_0 + \mathbf{e}_{1,k},$$
where $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_B}$, $\mathbf{J}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$ for $k \in \{\delta_1,\ldots,\delta_Q\}$ and $\mathbf{J}_k \overset{s}{\approx} \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $k \in [N]\backslash\{\delta_1,\ldots,\delta_Q\}$.
8. We require $\tau > s_1(\mathbf{J}_{\hat{i}}^*)$ in order to rely on ReRand algorithm for security proof. According to Lemma A.1, $s_1(\mathbf{J}_{\hat{i}}^*)$ is bounded by $1/\sqrt{2\pi}\cdot\sqrt{\rho^2+s^2}\cdot(\sqrt{t}+\sqrt{m}+\sqrt{\lambda})$.
9. We choose N, Q, w to satisfy the requirement for Cover-free Set (Lemma 2.7).

Our parameters may be chosen as: $n = \mathsf{poly}(\lambda)$, $m = (n+1)\log q$, $Q = O(\lambda)$, $w = \Theta(\lambda)$, $N = O(w\lambda^3)$, $s_B = \omega(\sqrt{\log n})$, $s = O(Ln\log q)^{O(\hat{d})}\cdot\omega(\sqrt{\log m})$, $\rho = O(Ln\log q)^{O(\hat{d})}\cdot\omega(\sqrt{\log m})\cdot\lambda^{\omega(1)}$, $\tau = \sqrt{\rho^2+s^2}\cdot(\sqrt{t}+\sqrt{m}+\sqrt{\lambda})$, $s_D = \rho\cdot m\cdot\omega(\sqrt{\log n})\cdot\lambda^{\omega(1)}$, $q = 2\sqrt{w}\cdot p^2\cdot s_D\cdot(t+2)$, where $L = \ell\cdot(n+1)^2\log q^2$, $\hat{d} = d\cdot O(\log m\log\log q)$.

Security.

Theorem (Restatement of Theorem 4.2) *Assuming the hardness of* LWE*, then the scheme described in Section 4.2 is a* $\mathsf{P\text{-}IPFE}$ *for the predicate class $\mathcal{F}$, message vector space $\mathcal{U}$ and key vector space $\mathcal{V}$, achieving* (Q,poly)-sel-*SIM security that allows up to Q 1-key pre-challenge query (and any polynomial number of 0-keys), according to Definition 2.*

Proof. We define a PPT simulator Sim and prove that for any PPT adversary $\mathcal{A}$, the ideal experiment with respect to Sim is computationally indistinguishable (under the LWE assumption) from the output of the real experiment.

Simulator. $\mathsf{QSim}^*(1^\lambda, 1^{|\mathbf{x}|}, 1^{|\mathbf{u}|})$:

1. $\mathsf{QSetup}^*(1^\lambda, 1^{|\mathbf{x}|}, 1^{|\mathbf{u}|})$: It generates all public parameters as in the real QSetup, except that it runs $(\mathbf{B}', \mathbf{T}_{\mathbf{B}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^m, q)$, then parse $\mathbf{B}' = \begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix}$, where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, and sets $\mathbf{B}$ be the public matrix in mpk. Then, it initializes $\mathsf{st} := \emptyset$.

2. $\mathsf{QKeyGen}^*_{\mathsf{pre}}(\mathsf{st}, f, \mathbf{v})$: It generates all secret keys as in the real $\mathsf{QKeyGen}$ algorithm and simultaneously maintains st that contains $\{f_{\hat{i}}, \mathbf{v}_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}}, \mathbf{K}_{\hat{i}} \cdot \mathbf{v}'_{\hat{i}})\}_{\hat{i}\in[Q']}$ for $f_{\hat{i}}$ such that $f_{\hat{i}}(\mathbf{x}^*) = 0$.
3. $\mathsf{QEnc}^*(\mathsf{st})$: It takes as input st that contains $d^{\mathsf{pre}}_{\hat{i}} = \langle \mathbf{u}^*, \mathbf{v}^{\mathsf{pre}}_{\hat{i}}\rangle$ if the adversary has queried for $(f_{\hat{i}}, \mathbf{v}_{\hat{i}})$ such that $f_{\hat{i}}(\mathbf{x}^*) = 0$ before the challenge query, then constructs the challenge ciphertext as follows.
   (a) It samples $\beta_0, \{\mathbf{c}_j\}_{j\in[L]}$ independently and uniformly from $\mathbb{Z}_q^m$.
   (b) Samples $\{\Psi_i, \Psi'_i\}_{i\in[\ell]}$ uniformly from $\mathbb{Z}_q^{(n+1)\times(n+1)\log q}$.
   (c) If $\mathsf{st} = \emptyset$, i.e., the adversary did not make any 1-key in the pre-challenge phase, it computes $\{\beta_{1,k}\}_{k\in[N]}$ as follows:
      – Choose Q random subset $(\Delta_1, \ldots, \Delta_Q)$ with size w according sampler $\mathsf{SamplerSet}(N, Q, w)$, sample $r_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_p$ for $\hat{i} \in [Q]$.
      – Generate random shares $\{r'_k\}_{k\in[N]}$ over $\mathbb{Z}_p$ under the following constraints: for $\hat{i} \in [Q]$, $\sum_{k\in\Delta_{\hat{i}}} r'_k = r_{\hat{i}}$. This can be done efficiently by the cover-freeness of the subsets, using the following standard procedure.
      Let $\delta_{\hat{i}}$ be a unique index that appears only in $\Delta_{\hat{i}}$ but not in the other subsets. To generate the random shares $\{r'_k\}_{k\in[N]}$, we first sample r'_k randomly for all $k \in [N]\backslash\{\delta_{\hat{i}}\}_{\hat{i}\in[Q]}$, and then fix $r'_{\delta_{\hat{i}}} = r_{\hat{i}} - \sum_{k\in\Delta_{\hat{i}}\backslash\{\delta_{\hat{i}}\}} r'_k$ for $\hat{i} \in [Q]$.
      – For $k \in [N]$, set $\mathbf{u}'_k = (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, 1)^\top \in \mathbb{Z}_p^{t+2}$ for $\widetilde{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^t$, sample $\widetilde{\beta}_k \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$.
      – Set $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}_{1,k} + \mathbf{u}'_k \mod q$.
   (d) Otherwise, if the adversary has submitted Q' 1-key queries in the pre-challenge phase, then update $\mathsf{st} = \mathsf{st} \parallel \{d^{\mathsf{pre}}_{\hat{i}} = \langle \mathbf{u}^*, \mathbf{v}^{\mathsf{pre}}_{\hat{i}}\rangle\}_{\hat{i}\in[Q']}$, then QEnc^* generates $\{\beta_{1,k}\}_{k\in[N]}$ to satisfy the decryption consistency as follows.
      – For $\hat{i} \in [Q']$, compute $\Psi_{f_{\hat{i}}} := \mathsf{HEval}_{f_{\hat{i}}}(\Psi)$. Let $\hat{f}_{\hat{i}}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_{f_{\hat{i}}}$, compute $\mathbf{H}_{\hat{f}_{\hat{i}},\Psi'} := \mathsf{MEvalFX}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}_{\hat{i}}, \Psi')$, $\mathbf{c}^\top_{\hat{f}_{\hat{i}}} := [\mathbf{c}_1^\top|\ldots|\mathbf{c}_L^\top] \cdot \mathbf{H}_{\hat{f}_{\hat{i}},\Psi'} - \underline{\Psi}_{f_{\hat{i}}}$.
      – Compute $\overline{u} \in \mathbb{Z}_p^t$ satisfying $\langle \widetilde{\mathbf{u}}, \mathbf{v}^{\mathsf{pre}}_{\hat{i}}\rangle = d^{\mathsf{pre}}_{\hat{i}} \mod p$ for $\hat{i} \in [Q']$.
      – Sample $Q - Q'$ random subsets of cardinality w using $\mathsf{SamplerSet}(N, Q, w)$, i.e. $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q'+1,Q]}$. By our setting of parameters, the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ are cover-free with an overwhelming probability.
      – For $\hat{i} \in [Q'+1, Q]$, sample $r_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_p$. Generate random shares $\{r'_k\}_{k\in[N]}$ over $\mathbb{Z}_p$ under the constraints that $\sum_{k\in\Delta_{\hat{i}}} r'_k = r_{\hat{i}}$ holds for $\hat{i} \in [Q]$, which also can be computed by the cover-freeness.
      – For $k \in [N]$, set $\mathbf{u}'_k = (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, 1)^\top \in \mathbb{Z}_p^{t+2}$.
      – Sample random vectors $\{\widetilde{\beta}_k\}_{k\in[N]}$ condition on the following equations:
      $$\sum_{k\in\Delta_{\hat{i}}} \widetilde{\beta}_k = \mathbf{K}^\top_{\hat{i}} \cdot \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\hat{f}_{\hat{i}}} \end{pmatrix} \text{ for } \hat{i} \in [Q'].$$
      – Sample $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ for $k \in [N]$, Set $\beta_{1,k} = \widetilde{\beta}_k + \mathbf{e}_{1,k} + \mathbf{u}'_k \mod q$.
   (e) It outputs the simulated ciphertext
   $$\mathsf{ct}^* := (\Psi, \Psi', \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]}).$$
4. $\mathsf{QKeyGen}^*_{\mathsf{post}}(\mathsf{st}, f, \mathbf{v})$ generates as in the real $\mathsf{QKeyGen}$ algorithm for all 0-key queries. Otherwise, assume that the current state contains $Q'(< Q)$ tuples of $f_{\hat{i}}, \mathbf{v}_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}}, \mathbf{K}_{\hat{i}} \cdot \mathbf{v}'_{\hat{i}})$ for $f_{\hat{i}}$, for a 1-key query $(f_{\hat{i}_p}, \mathbf{v}_{\hat{i}_p})$, the simulator computes as follows.
   – Set $\Delta = \Delta_{\hat{i}_p}$ for which is chosen during QEnc^* algorithm.

– Compute $\mathbf{P}_\Delta = \sum_{k\in\Delta} \mathbf{P}_k$ and $\widetilde{\beta}_\Delta = \sum_{k\in\Delta} \widetilde{\beta}_k$, where $\{\widetilde{\beta}_k\}_{k\in[N]}$ are chosen during QEnc^*.
– Compute $\Psi_f \coloneqq \mathsf{HEval}_f(\Psi)$, $\mathbf{H}_{\hat{f}}$ and $\mathbf{H}_{\hat{f},\psi'}$, and use these results to compute $\mathbf{B}_{\hat{f}}$ and $\mathbf{c}_{\hat{f}}$, respectively.
– Sample $\mathbf{J}_{\hat{i}_p} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$, use $\mathbf{T}_{\mathbf{B}'}$ to sample $\begin{bmatrix}\mathbf{K}_{\hat{i}_p,1}\\ \mathbf{K}_{\hat{i}_p,2}\end{bmatrix}$ by $\mathsf{SampleLeft}$ such that

$$\begin{bmatrix}\mathbf{B} & \mathbf{B}_{\hat{f}_{\hat{i}_p}}\\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}_{\hat{i}_p}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{K}_{\hat{i}_p,1}\\ \mathbf{K}_{\hat{i}_p,2}\end{bmatrix} = \begin{bmatrix}\mathbf{P}_\Delta\\ \widetilde{\beta}_\Delta^\top\end{bmatrix} - \begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix} \cdot \mathbf{J}_{\hat{i}_p}.$$

– Set $\mathbf{K}_{f_{\hat{i}_p}} = \begin{bmatrix}\mathbf{J}_{\hat{i}_p} + \mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$.
– Given $d^{\mathsf{post}} = \langle \mathbf{u}^*, \mathbf{v}\rangle$, compute $\theta = d^{\mathsf{post}} - \langle \widetilde{\mathbf{u}}, \mathbf{v}\rangle$ and set $\mathbf{v}' = (\mathbf{v}, 1, \theta + r)$ for $r \coloneqq r_{\hat{i}}$.
– Output $\mathsf{sk}_{f,\mathbf{v}} \coloneqq (\Delta, \mathbf{v}', \mathbf{K}_f \cdot \mathbf{v}')$.

Auxiliary Algorithms.

$\mathsf{QSetup}_1^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Do the following:

1. Generate $(\mathbf{B}, \mathbf{T}_\mathbf{B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
2. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, compute $\mathbf{z}^\top \coloneqq \mathbf{s}^\top\mathbf{B} + \mathbf{e}_0^\top$.
3. Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ for $i \in [\ell]$ and compute

$$\Psi_i' \coloneqq \begin{pmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{pmatrix}\mathbf{R}_i + x_i^*\mathbf{G}.$$

   Let $\psi' = (\psi_1', \ldots, \psi_L')$ denote the bit-representation of $\Psi \coloneqq [\Psi_1|\cdots|\Psi_\ell]$.
4. Set $\mathbf{B}_j = \mathbf{B}\cdot\mathbf{W}_j - \psi_j \cdot \overline{\mathbf{G}}$ for $j \in [L]$, where $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ for $j \in [L]$.
5. Choose Q random subsets $(\Delta_1, \ldots, \Delta_Q)$ with cardinality w according sampler $\mathsf{SamplerSet}(N, Q, w)$. By cover-freeness, for every $\hat{i} \in [Q]$, there exists a unique index $\delta_{\hat{i}}$ that only appears in $\mathrm{ALS}_{\hat{i}}$ but not the other subsets.
6. Sample $\mathbf{J}_{\hat{i}}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $\hat{i} \in [Q]$, and sample $\mathbf{P}_k \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ for $k \in [N]$ under the constraint $\sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_{\hat{i}}^*$. Denote $\sum_{k\in\Delta_{\hat{i}}} \mathbf{P}_k$ as $\mathbf{P}_{\Delta_{\hat{i}}}$.
7. Sample $r_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_p$ for $\hat{i} \in [Q]$. Generate random shares $\{r_k'\}_{k\in[N]}$ over $\mathbb{Z}_p$ under the constraints that $\sum_{k\in\Delta_{\hat{i}}} r_k' = r_{\hat{i}}$ holds for $\hat{i} \in [Q]$, which also can be computed by the cover-freeness.
8. Output the public and master secret keys.

$$\mathsf{mpk} \coloneqq (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \{\mathbf{P}_k\}_{k\in[N]}),$$
$$\mathsf{msk} \coloneqq (\mathbf{T}_\mathbf{B}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{W}_j\}_{j\in[L]}, \{\mathbf{J}_{\hat{i}}^*, r_{\hat{i}}\}_{\hat{i}\in[Q]}, \mathbf{s}).$$

$\mathsf{QEnc}_1^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mathbf{u}^*)$: Do the following:

1. Set $\beta_0 \coloneqq \mathbf{z}$.
2. For $j \in [L]$, compute $\mathbf{c}_j \coloneqq \mathbf{W}_j^\top\beta_0$, where $\mathbf{W}_j$ are the matrices in the msk generated by Setup_1^*.
3. $\{\beta_{1,k}\}_{k\in[N]}$ is computed as real encryption algorithm, except that the secret randomness $\mathbf{s}$ is set the one chosen in Setup_1^*.
4. Output the ciphertext $\mathsf{ct}^* \coloneqq (\Psi, \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]})$.

$\mathsf{QKeyGen}_1^*(\mathsf{msk}, \mathsf{st}, f, \mathbf{v})$: This algorithm is stateful that keeps track of how many keys have been queried before. Particularly, it does the following:

1. Let $\hat{f}$ denote the circuit computing $\Psi \mapsto \overline{\Psi}_f$, compute the homomorphic public key corresponding to circuit $\hat{f}$ as

$$\mathbf{H}_{\hat{f}} \coloneqq \mathsf{MEvalF}(\{\mathbf{B}_j\}_{j\in[L]}, \hat{f}),$$

$$\begin{aligned}\mathbf{B}_{\hat{f}} &\coloneqq [\mathbf{B}_1 \mid \cdots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{f}} \\ &= [\mathbf{B}_1 + \psi_1'\overline{\mathbf{G}} \mid \cdots \mid \mathbf{B}_L + \psi_L'\overline{\mathbf{G}}] \cdot \mathbf{H}_{\hat{f},\psi'} - \overline{\Psi}'_f \\ &= \mathbf{B}[\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi'} - \overline{\Psi}'_f \\ &= \mathbf{B}(\mathbf{W}_{\hat{f}} - \mathbf{R}_f) - f(\mathbf{x}^*)\overline{\mathbf{G}}\end{aligned}$$

   where $\mathbf{W}_{\hat{f}} \coloneqq [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_L] \cdot \mathbf{H}_{\hat{f},\psi'}$, $\Psi'_f = \begin{pmatrix}\mathbf{B}\\ \mathbf{z}'^\top\end{pmatrix}\mathbf{R}_f + f(\mathbf{x}^*)\mathbf{G}$.

2. For 0-key query $(f, \mathbf{v})$ such that $f(\mathbf{x}^*) \neq 0$, firstly sample a randomness $r \xleftarrow{\$} \mathbb{Z}_p$ and a fresh random subset $\Delta \subseteq [N]$ with cardinality w according sampler $\mathsf{SamplerSet}(N, Q, w)$, then generate

$$\mathbf{K}_f \leftarrow \mathsf{SampleRight}(\mathbf{B}, \mathbf{G}, \mathbf{W}_{\hat{f}} - \mathbf{R}_f, \textstyle\sum_{k\in\Delta}\mathbf{P}_k, s),$$

   satisfying $[\mathbf{B}|\mathbf{B}_{\hat{f}}] \cdot \mathbf{K}_f = \mathbf{P}_\Delta = \sum_{k\in\Delta}\mathbf{P}_k$.

3. For 1-key query $(f_{\hat{i}}, \mathbf{v}_{\hat{i}})$ such that $f_{\hat{i}}(\mathbf{x}^*) = 0$, the algorithm does the following. We use index $\hat{i} \in [Q]$ to denote the number of overall 1-key queries currently.
   - Set $\Delta \coloneqq \Delta_{\hat{i}}$ and $r \coloneqq r_{\hat{i}}$. Notice that $\Delta_{\hat{i}}, r_{\hat{i}}$ and $\mathbf{J}^*_{\hat{i}}$ are all sampled during the QSetup_1^*.
   - Sample $\mathbf{K}_{\hat{i},2} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times(t+2)},s}$, and set

$$\mathbf{K}_{f_{\hat{i}}} \coloneqq \begin{bmatrix}\mathbf{J}^*_{\hat{i}} - (\mathbf{W}_{\hat{f}_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}}) \cdot \mathbf{K}_{\hat{i},2} \\ \mathbf{K}_{\hat{i},2}\end{bmatrix}.$$

   Then, by the construction, we have

$$[\mathbf{B}|\mathbf{B}_{\hat{f}_{\hat{i}}}] \cdot \mathbf{K}_f = [\mathbf{B}|\mathbf{B}(\mathbf{W}_{\hat{f}_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}})] \cdot \mathbf{K}_f = \mathbf{B}\mathbf{J}^*_{\hat{i}} = \sum_{k\in\Delta}\mathbf{P}_k.$$

4. Set $\mathbf{v}' = (\mathbf{v}^\top, 1, r)^\top$.
5. Return $\mathsf{sk}_{f,\mathbf{v}} = (\Delta, \mathbf{v}', \mathbf{K}_f \cdot \mathbf{v}')$ and update $\mathsf{st} \coloneqq \mathsf{st} \cup (f_{\hat{i}}, \mathbf{v}_{\hat{i}}, \mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}}, \mathbf{K}_{\hat{i}} \cdot \mathbf{v}'_{\hat{i}}))$ if $f(\mathbf{x}^*) = 0$.

$\mathsf{QKeyGen}_2^*(\mathsf{msk}, \mathsf{st}, f, \mathbf{v})$: Same as $\mathsf{QKeyGen}_1^*$, except for the way of generating $\mathbf{v}'$ for post-challenge 1-key queries. Note that in the post-challenge phase, the challenger has access to $d^{\mathsf{post}} = \langle \mathbf{u}^*, \mathbf{v}\rangle$ for key query $(f, \mathbf{v})$ where $f(\mathbf{x}^*) = 0$, and $\{d^{\mathsf{pre}}_{\hat{i}} = \langle \mathbf{u}^*, \mathbf{v}_{\hat{i}}\rangle\}_{\hat{i}\in[Q']}$ for Q' pre-challenge key queries. Specifically, if the adversary made no 1-key query in the pre-challenge phase, i.e., $Q' = 0$, then the challenger samples a random $\widetilde{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^t$. Otherwise, the challenger computes $\widetilde{\mathbf{u}} \in \mathbb{Z}_p^t$ satisfying $\langle \widetilde{\mathbf{u}}, \mathbf{v}_{\hat{i}}\rangle \mod p$ for all $\hat{i} \in [Q']$. Next, the challenger computes $\theta = d^{\mathsf{post}} - \langle \widetilde{\mathbf{u}}, \mathbf{v}\rangle \mod p$ and set $\mathbf{v}' = (\mathbf{v}, 1, \theta + r)$ for $r \coloneqq r_{\hat{i}}$.

$\mathsf{QSetup}_2^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Same as QSetup_1^*, except that $\mathbf{B}$ is sampled uniformly from $\mathbb{Z}_q^{n\times m}$.

$\mathsf{QEnc}_2^*(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mathbf{u}^*)$: Same as in QEnc_1^*, except that each $\mathbf{u}'_k$ is set as $(\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, \frac{1}{w})^\top \in \mathbb{Z}_p^{t+2}$, where r'_k is sampled during QSetup_2^* and $\widetilde{\mathbf{u}}$ is computed depending on whether there are pre-challenge 1-key queries. Assume that the adversary has made Q' 1-key queries before the challenge phase, then the challenger samples $\widetilde{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^t$ if $Q' = 0$, otherwise the challenger computes $\widetilde{\mathbf{u}}$ such that $\langle \widetilde{\mathbf{u}}, \mathbf{v}_{\hat{i}}\rangle = d^{\mathsf{pre}}_{\hat{i}} \mod p$ for $\hat{i} \in [Q']$.

$\mathsf{QSetup}_3^*(1^\lambda, 1^{|\mathbf{u}|}, \mathbf{x}^*)$: Same as QSetup_2^*, except that $\{\mathbf{P}_k\}_{k\in[N]}$ are chosen as follows.

1. Sample $\mathbf{J}^*_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $\hat{i} \in [Q]$, as in QSetup^*_2.
2. Sample $\mathbf{J}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $k \in [N]\backslash\{\delta_1,\ldots,\delta_Q\}$ and set $\mathbf{J}_{\delta_{\hat{i}}} = \mathbf{J}^*_{\hat{i}} - \sum_{k\in\Delta^*_{\hat{i}}\backslash\{\delta_{\hat{i}}\}} \mathbf{J}_k$ for $\hat{i} \in [Q]$, then we have $\mathbf{J}^*_{\hat{i}} = \sum_{k\in\Delta_{\hat{i}}} \mathbf{J}_k$.
3. Set $\mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_k$ for $k \in [N]$.

$\mathsf{QEnc}^*_3(\mathsf{mpk},\mathsf{msk},\mathsf{st})$: Same as QEnc^*_2, except the way of generating $\beta_{1,k}$. Specifically, the challenge computes $\beta_{1,k} = \mathbf{J}_k^\top\cdot\beta_0 + \mathbf{e}_{1,k} + \mathbf{u}'_k \mod q$ for $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^{t+2},s_D}$ and $\mathbf{u}'_k$ chosen as in QEnc^*_2.
$\mathsf{QSetup}^*_4(1^\lambda,1^{|\mathbf{u}|},\mathbf{x}^*)$: Same as QSetup^*_3, except that the challenger samples $\mathbf{z}' \xleftarrow{\$} \mathbb{Z}_q^m$.
$\mathsf{QEnc}^*_4(\mathsf{mpk},\mathsf{msk},\mathsf{st})$: Same as QEnc^*_3, except that the challenger samples $\mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^m$.
$\mathsf{QSetup}^*_5(1^\lambda,1^{|\mathbf{u}|},\mathbf{x}^*)$: Same as QSetup^*_4, except for generating $\mathbf{B}$ and $\mathbf{z}$ as follows:

1. Generate $(\mathbf{B}',\mathbf{T}_{\mathbf{B}'}) \leftarrow \mathsf{TrapGen}(1^{n+1},1^m,q)$, then parse $\mathbf{B}'$ as $\begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix}$.
2. Define $\widetilde{\beta}_k = \mathbf{J}_k^\top\cdot\mathbf{z}$ for $k\in[N]$.
3. Set $\mathbf{B}$ as the public matrix in mpk.
4. Compute remaining elements as in QSetup^*_4. Additionally, add $\{\widetilde{\beta}_k\}_{k\in[N]}$ into msk.

$\mathsf{QEnc}^*_5(\mathsf{mpk},\mathsf{msk},\mathsf{st})$: Same as QEnc^*_4, except that $\beta_{1,k}$ is computed as $\widetilde{\beta}_k + \mathbf{e}_{1,k} + \mathbf{u}'_k \mod q$.
$\mathsf{QSetup}^*_6(1^\lambda,1^{|\mathbf{u}|},\mathbf{x}^*)$: Same as QSetup^*_5, except for generating $\mathbf{P}_k$ and $\widetilde{\beta}_k$ as follows:

1. Sample $\mathbf{P}_k$ randomly from $\mathbb{Z}_q^{m\times n}$ under the constraint that $\sum_{k\in\Delta_{\hat{i}}}\mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}^*_{\hat{i}}$, which is thus distributed exactly the same as in $\mathsf{QPE.Setup}^*_1$.
2. Sample $\widetilde{\beta}_k$ randomly from $\mathbb{Z}_q^m$ for under the constraint that $\sum_{k\in\Delta_{\hat{i}}}\widetilde{\beta}_k = \mathbf{J}^*_{\hat{i}}\cdot\mathbf{z}$, denote $\sum_{k\in\Delta_{\hat{i}}}\widetilde{\beta}_k$ as $\widetilde{\beta}_{\Delta_{\hat{i}}}$.

It is important to note that the generation of $\{\mathbf{J}_k\}_{k\in[N]}$ is no longer required in the QSetup^*_6 algorithm.
$\mathsf{QKeyGen}^*_3(\mathsf{msk},\mathsf{st},f,\mathbf{v})$: Do the following:

1. For 0-key query, generate and return the key $\mathsf{sk}_{f,\mathbf{v}}$ as in $\mathsf{QKeyGen}^*_2$.
2. For 1-key query, let $f_{\hat{i}}$ be the $\hat{i}$-th 1-key query, set $\mathbf{v}'_{\hat{i}}$ and $\Delta = \Delta_{\hat{i}}$ as in $\mathsf{QKeyGen}^*_2$.
   – Sample $\mathbf{J}_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$. Use $\mathbf{T}_{\mathbf{B}'}$ to sample $\begin{bmatrix}\mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$ by $\mathsf{SampleLeft}$ such that
   $$\begin{bmatrix}\mathbf{B} & \mathbf{B}_{\hat{f}}\\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix} = -\begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix}\cdot\mathbf{J}_{\hat{i}} + \begin{bmatrix}\mathbf{P}_\Delta\\ \widetilde{\beta}_\Delta^\top\end{bmatrix}.$$
   – Set $\mathbf{K}_f = \begin{bmatrix}\mathbf{J}_{\hat{i}} + \mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$.
   – Return $\mathsf{sk}_{f,\mathbf{v}} := (\Delta,\mathbf{v}',\mathbf{K}_f\cdot\mathbf{v}')$.

$\mathsf{QSetup}^*_7(1^\lambda,1^{|\mathbf{u}|},1^{|\mathbf{x}|})$: Same as QSetup^*_6, except that sample $\{\mathbf{B}_j\}$ and $\{\mathbf{P}_k\}$ from random as in the normal $\mathsf{QPE.Setup}$, as well as sample $\widetilde{\beta}_k \xleftarrow{\$} \mathbb{Z}_q^m$ for $k\in[N]$.
$\mathsf{QEnc}^*_6(\mathsf{mpk},\mathsf{msk},\mathsf{st})$: Same as QEnc^*_5, except that sample $\{\mathbf{c}_j\}_{j\in[L]}$ and Ψ,Ψ' randomly.

Hybrids.

$\mathcal{H}_0$: The real experiment.

$\mathcal{H}_1$: The real game algorithms QSetup and QEnc are replaced with QSetup^*_1 and QEnc^*_1, which use the knowledge of $\mathbf{x}^*$ to generate the public parameters, the master public/secret keys, and additionally samples random $\mathbf{P}_k$ under the constrain $\sum_{k\in \mathrm{ALS}_{\hat{i}}} \mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}^*_{\hat{i}}$. $\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically close by an application of the Leftover Hash Lemma.

$\mathcal{H}_2$: The real game algorithm $\mathsf{QKeyGen}$ is replaced with $\mathsf{QKeyGen}^*_1$ where instead of using the trapdoor $\mathbf{T_B}$, secret keys for a 0-key queries are sampled using the public trapdoor $\mathbf{T_G}$ along with the trapdoor information generated in QSetup^*_1, and the secret key for the $\hat{i}$-th 1-key query for function $(f_{\hat{i}}, \mathbf{v}_{\hat{i}})$ is generated as $\left(\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}} = (\mathbf{v}^\top_{\hat{i}}, 1, r_{\hat{i}})^\top, \begin{bmatrix}\mathbf{J}^*_{\hat{i}} - (\mathbf{W}_{f_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}})\cdot\mathbf{K}_{\hat{i},2} \\ \mathbf{K}_{\hat{i},2}\end{bmatrix}\cdot\mathbf{v}'_{\hat{i}}\right)$.

$\mathcal{H}_3$: $\mathsf{QKeyGen}^*_1$ is replaced with $\mathsf{QKeyGen}^*_2$, in which the randomness component encoded in each $\mathbf{v}'$ is computed differently.

$\mathcal{H}_4$: QSetup^*_1 is replaced by QSetup^*_2, in which $\mathbf{B}$ is sampled randomly.

$\mathcal{H}_5$: QEnc^*_1 is replaced by QEnc^*_2, in which the message vector $\mathbf{u}'_k$ is computed using randomness values $\{r_k\}$ sampled during QSetup^*_2, along with additional information. Note that the challenger computes $\widetilde{\mathbf{u}}$ in the same manner as in $\mathsf{QKeyGen}^*_2$. Therefore, the computation of $\widetilde{\mathbf{u}}$ can be viewed as being transferred from $\mathsf{QKeyGen}^*_2$ to QEnc^*_2, following the same generation approach.

$\mathcal{H}_6$: QSetup^*_2 is replaced by QSetup^*_3, in which the public matrices $\{\mathbf{P}_k\}$ are generated by first sampling matrices $\mathbf{J}_k$ from Gaussian distributions, then setting $\mathbf{P}_k = \mathbf{B}\cdot\mathbf{J}_k$.

$\mathcal{H}_7$: QEnc^*_2 is replaced by QEnc^*_3, in which $\beta_{1,k}$ is computed using β_0 and $\mathbf{J}_k$.

$\mathcal{H}_8$: QSetup^*_3 is replaced by QSetup^*_4, in which $\mathbf{z}'$ is chosen from uniformly random and thus public matrices $\{\mathbf{B}_j\}_{j\in[L]}$ are derived from it.

$\mathcal{H}_9$: QEnc^*_3 is replaced by QEnc^*_4, in which $\mathbf{z}$ is chosen from uniformly random and thus ciphertext elements $(\beta_0, \{\beta_{1,k}\}_{k\in[N]}, \Psi', \{\mathbf{c}_j\}_{j\in[L]})$ are derived from it.

$\mathcal{H}_{10}$: QSetup^*_4 and QEnc^*_4 are replaced by QSetup^*_5 and QEnc^*_5. In QSetup^*_5, the $\mathsf{TrapGen}$ algorithm outputs the public matrix $\mathbf{B}' = \begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix}$ together with $\mathbf{T}_{\mathbf{B}'}$, the vector $\mathbf{z}$ is set as last row of output matrix $\mathbf{B}'$ from $\mathsf{TrapGen}$ algorithm instead of sampling uniformly and the vectors $\{\widetilde{\beta}_k\}_{k\in[N]}$ are added into the master secret key. QEnc^*_5 is almost the same as the QEnc^*_4 except that $\beta_{1,k}$ is computed using $\widetilde{\beta}_k$.

$\mathcal{H}_{11}$: QSetup^*_5 is replaced by QSetup^*_6, in which $\mathbf{P}_k$ and $\widetilde{\beta}_k$ are instead sampled randomly under specific conditions.

$\mathcal{H}_{12}$: $\mathsf{QKeyGen}^*_2$ is replaced with $\mathsf{QKeyGen}^*_3$, in which the response to 1-key query is generated using the trapdoor $\mathbf{T}_{\mathbf{B}'}$ such that

$$\begin{bmatrix}\mathbf{B} & \mathbf{B}_{\hat{f}} \\ \mathbf{z}^\top & \mathbf{c}_{\hat{f}}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{P}_\Delta \\ \widetilde{\beta}^\top_\Delta\end{bmatrix} - \begin{bmatrix}\mathbf{B}\\ \mathbf{z}^\top\end{bmatrix}\cdot\mathbf{J}_{\hat{i}}.$$

$\mathcal{H}_{13}$: QSetup^*_6 and QEnc^*_5 are replaced by QSetup^*_7 and QEnc^*_6. In particular, the public matrices $\{\mathbf{B}_j\}, \{\mathbf{P}_k\}$ are generated as the real world, and $\{\widetilde{\beta}_k\}_{k\in[N]}$ are sampled uniformly at random. Also the ciphertext components $\{\mathbf{c}_j\}_{j\in[L]}$ and Ψ, Ψ' are sampled uniformly.

$\mathcal{H}_{14}$: The ideal experiment.

Next, we will prove that each pair of adjacent hybrid arguments is indistinguishable.

Lemma C.12 *$\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.11 and is therefore omitted for brevity.

Lemma C.13 *$\mathcal{H}_1$ and $\mathcal{H}_2$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.12 and is therefore omitted for brevity.

Lemma C.14 *$\mathcal{H}_2$ and $\mathcal{H}_3$ are equivalent.*

Proof. The only difference of $\mathcal{H}_2$ and $\mathcal{H}_3$ lies in how the vector $\mathbf{v}'$ is set for post-challenge 1-key queries. Specifically, we set $\mathbf{v}_{\hat{i}}$ as $(\mathbf{v}^\top_{\hat{i}}, 1, r_{\hat{i}})$ in $\mathcal{H}_2$, and as $(\mathbf{v}^\top_{\hat{i}}, 1, \theta_{\hat{i}} + r_{\hat{i}})$ in $\mathcal{H}_3$. Since each $r_{\hat{i}}$ is chosen as random over $\mathbb{Z}_p$ and $\theta_{\hat{i}}$ is computed independently for $\hat{i} \in [Q']$, the resulting $\theta_{\hat{i}} + r_{\hat{i}}$) is still distributed as uniformly random in $\mathbb{Z}_p$. Therefore, the distribution of $\mathbf{v}'$ in two hybrids are identically distributed. Furthermore, notice that each message vector $\mathbf{u}'_k$ is of the form $(\frac{1}{w}\mathbf{u}^{*\top}, 0, 0)$, the decryption outputs stay consistent. Thus, we have $\mathcal{H}_2 \equiv \mathcal{H}_3$. □

□

Lemma C.15 *$\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma C.3 and is therefore omitted for brevity.

Lemma C.16 *$\mathcal{H}_4$ and $\mathcal{H}_5$ are computationally indistinguishable assuming the security of N-ALS scheme.*

Proof. The difference between $\mathcal{H}_4$ and $\mathcal{H}_5$ lies in how the message vector $\mathbf{u}'_k$ is computed. We reduce the distinguishing advantage of $\mathcal{H}_4$ and $\mathcal{H}_5$ to the security of N-ALS scheme.

On receiving the public key $(\mathbf{A}_{\text{ALS}}, \{\mathbf{D}_{\text{ALS},k}\}_{k\in[N]}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{(t+2)\times n}$ from the N-ALS challenger (as described in Appendix D.2), we simulate the view of the distinguisher for $\mathcal{H}_4$ versus $\mathcal{H}_5$ as follows.

– $\mathsf{Setup}(1^\lambda, 1^\ell, 1^d, 1^Q, \mathbf{x}^*)$:
  1. Set $\mathbf{B} \coloneqq \mathbf{A}_{\text{ALS}}^\top$.
  2. Sample $\mathbf{s}' \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}'_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_B}$, compute $\mathbf{z}'^\top \coloneqq \mathbf{s}'^\top \mathbf{B} + \mathbf{e}_0'^\top$.
  3. Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{m\times m}$ for $i \in [\ell]$ and compute $\Psi'_i \coloneqq \begin{pmatrix} \mathbf{B} \\ \mathbf{z}'^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G}$. Let $\psi'_1, \ldots, \psi'_L$ denote the bit-representation of $\Psi' \coloneqq [\Psi'_1 | \cdots | \Psi'_\ell]$.
  4. Set $\mathbf{B}_j = \mathbf{B} \cdot \mathbf{W}_j - \psi'_j \cdot \overline{\mathbf{G}}$ for $j \in [L]$, where $\mathbf{W}_j \xleftarrow{\$} \{-1,1\}^{m\times m}$.
  5. Sample $\mathbf{J}^*_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times t}, \sqrt{\rho^2+s^2}}$ for $\hat{i} \in [Q]$.
  6. Choose Q random subsets $(\Delta_1, \ldots, \Delta_Q)$ with cardinality w according sampler $\mathsf{SamplerSet}(N, Q, w)$. By cover-freeness, for every $\hat{i} \in [Q]$, there exists a unique index $\delta_{\hat{i}}$ that only appears in $\Delta_{\hat{i}}$ but not the other subsets.
  7. For $k \in \{\delta_1, \ldots, \delta_Q\}$, set $\mathbf{P}_k \coloneqq \mathbf{D}_{\text{ALS},k}^\top + \mathbf{B}\mathbf{J}^*_{\hat{i}}$, otherwise, set $\mathbf{P}_k \coloneqq \mathbf{D}_{\text{ALS},k}^\top$.
  8. Sample $r_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_p$ for $\hat{i} \in [Q]$. Generate random shares $\{r'_k\}_{k\in[N]}$ over $\mathbb{Z}_p$ under the constraints that $\sum_{k\in\Delta_{\hat{i}}} r'_k = r_{\hat{i}}$ holds for $\hat{i} \in [Q]$.
  9. Set and return $\mathsf{mpk} = (\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \{\mathbf{P}_k\}_{k\in[N]})$ to the distinguisher.

– $\mathcal{O}\mathsf{KeyGen}(\mathsf{msk}, \mathsf{st}, f, \mathbf{v})$: For key query $(f, \mathbf{v})$,
  - For the case where $f(\mathbf{x}^*) \neq 0$, generate the secret keys using $\mathbf{T}_{\mathbf{G}}$, as in $\mathcal{H}_4$.
  - For the case where $f_{\hat{i}}(\mathbf{x}^*) = 0$, the challenger sets $\mathbf{v}'_{\hat{i}}$ and $\mathbf{v}_{\text{ALS},\hat{i},k}$ as follows:

$$\mathbf{v}'_{\hat{i}} = \begin{cases} (\mathbf{v}_{\hat{i}}^\top, 1, r_{\hat{i}})^\top & \text{for pre-challenge query} \\ (\mathbf{v}_{\hat{i}}^\top, 1, \theta_{\hat{i}} + r_{\hat{i}})^\top & \text{for post-challenge query} \end{cases}$$

$$\mathbf{v}_{\text{ALS},\hat{i},k} = \begin{cases} \mathbf{v}'_{\hat{i}} & \text{for } k \in \Delta_{\hat{i}} \\ \mathbf{0} & \text{for } k \in [N]\backslash\Delta_{\hat{i}} \end{cases}$$

    where $r_{\hat{i}}$ is chosen during Setup^*, $\theta_{\hat{i}}$ is computed as in $\mathsf{QKeyGen}_2^*$.

    Next, the challenger submits the key query $\{\mathbf{v}_{\text{ALS},\hat{i},k}\}_{k\in[N]}$ to the N-ALS challenger, receiving $\mathsf{isk}_{\hat{i}}$ in response. Next, compute

$$\mathbf{k}_{\hat{i}} \coloneqq \begin{bmatrix} \mathsf{isk}_{\mathbf{v}'} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{J}^*_{\hat{i}} - (\mathbf{W}_{f_{\hat{i}}} - \mathbf{R}_{f_{\hat{i}}}) \cdot \mathbf{K}_{\hat{i},2} \\ \mathbf{K}_{\hat{i},2} \end{bmatrix} \cdot \mathbf{v}'_{\hat{i}},$$

where $\mathbf{K}_{\hat{i},2} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times(t+2)},s}$. Return $\mathsf{sk}_{\hat{i}} = (\Delta_{\hat{i}}, \mathbf{v}'_{\hat{i}}, \mathbf{k}_{\hat{i}})$.

– $\mathcal{O}\mathsf{Enc}(\mathsf{mpk}, \mathsf{msk}, \mathsf{st}, \mathbf{u}^*)$:

- Set $\mathbf{u}_{\mathrm{ALS},0,k} = (\frac{1}{w}\mathbf{u}^{*\top}, 0, 0)^\top$.
- Set $\mathbf{u}_{\mathrm{ALS},1,k} = (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, \frac{1}{w})^\top$, where $\widetilde{\mathbf{u}}$ is chosen as random if the adversary made no 1-key in the pre-challenge phase, otherwise is computed to satisfy $\langle \widetilde{\mathbf{u}}, \mathbf{v}_{\hat{i}} \rangle = d_{\hat{i}}^{\mathsf{pre}} \mod p$ for pre-challenge 1-key queries $\hat{i} \in [Q']$.
- Send the challenge query $(\{\mathbf{u}_{\mathrm{ALS},0,k}, \mathbf{u}_{\mathrm{ALS},1,k}\}_{k\in[N]})$ to the N-ALS challenger. When receiving $(\mathsf{ict}_0, \mathsf{ict}_{1,k})$, compute the challenge ciphertext as follows:

$$\beta_0 \coloneqq \mathsf{ict}_0,\ \mathbf{c}_j \coloneqq \mathbf{W}_j^\top \beta_0,$$
$$\Psi_i = \begin{pmatrix} \mathbf{B} \\ \mathbf{z}^\top \end{pmatrix} \mathbf{R}_i + x_i^* \mathbf{G},\ \text{where } \mathbf{z} \coloneqq \beta_0,$$

$$\beta_{1,k} = \begin{cases} \mathsf{ict}_{1,k} + \mathbf{e}_{1,k} & \text{for } k \in [N]\backslash\{\delta_1, \ldots, \delta_Q\} \\ \mathsf{ReRand}(\mathsf{ict}_0, \mathbf{J}^*_{\hat{i}}, \sigma_{\mathrm{ALS}}, \tau) + \mathsf{ict}_{1,k} + \mathbf{e}_{1,k} & \text{for } k \in \{\delta_1, \ldots, \delta_Q\} \end{cases}$$

- Return $\mathsf{ct}^* \coloneqq (\Psi, \Psi', \beta_0, \{\beta_{1,k}\}_{k\in[N]}, \{\mathbf{c}_j\}_{j\in[L]})$.

We claim that all the queries submitted to the N-ALS challenger are admissible. First, notice that we have

$$\begin{aligned} &\sum_{k\in[N]} \langle \mathbf{v}_{\mathrm{ALS},\hat{i},k}, \mathbf{u}_{\mathrm{ALS},0,k} \rangle \\ =& \sum_{k\in\Delta_{\hat{i}}} \langle (\mathbf{v}_{\hat{i}}^\top, 1, r_{\hat{i}})^\top, (\frac{1}{w}\mathbf{u}^*, 0, 0) \rangle \\ =& \langle \mathbf{v}_{\hat{i}}, \mathbf{u}^* \rangle. \end{aligned}$$

For key queries that transformed from pre-challenge queries, we have

$$\begin{aligned} &\sum_{k\in[N]} \langle \mathbf{v}_{\mathrm{ALS},\hat{i},k}, \mathbf{u}_{\mathrm{ALS},1,k} \rangle \\ =& \sum_{k\in\Delta_{\hat{i}}} \langle (\mathbf{v}_{\hat{i}}^\top, 1, r_{\hat{i}})^\top, (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, \frac{1}{w}) \rangle \\ =& (\mathbf{v}_{\hat{i}}^\top, 1, r_{\hat{i}}) \cdot (\widetilde{\mathbf{u}}^\top, -\sum_{k\in\Delta_{\hat{i}}} r'_k, 1) \\ =& \langle \mathbf{v}_{\hat{i}}, \widetilde{\mathbf{u}} \rangle \end{aligned}$$

The last equations is ensured by the choices of r'_k, which satisfies $\sum_{k\in\Delta_{\hat{i}}} r'_k = r_{\hat{i}}$. Under the constraint for computing $\widetilde{\mathbf{u}}$, it holds that $\langle \mathbf{v}_{\hat{i}}, \widetilde{\mathbf{u}} \rangle = \langle \mathbf{v}_{\hat{i}}, \mathbf{u}^* \rangle$. Similarly, we have the following relationship for queries that obtained from post-challenge 1-key query.

$$\begin{aligned} &\sum_{k\in[N]} \langle \mathbf{v}_{\mathrm{ALS},\hat{i},k}, \mathbf{u}_{\mathrm{ALS},1,k} \rangle \\ =& \sum_{k\in\Delta_{\hat{i}}} \langle (\mathbf{v}_{\hat{i}}^\top, 1, \theta_{\hat{i}} + r_{\hat{i}})^\top, (\frac{1}{w}\widetilde{\mathbf{u}}^\top, -r'_k, \frac{1}{w}) \rangle \\ =& (\mathbf{v}_{\hat{i}}^\top, 1, \theta_{\hat{i}} + r_{\hat{i}}) \cdot (\widetilde{\mathbf{u}}^\top, -\sum_{k\in\Delta_{\hat{i}}} r'_k, 1) \\ =& \langle \mathbf{v}_{\hat{i}}, \widetilde{\mathbf{u}} \rangle + \theta_{\hat{i}} = \langle \mathbf{v}_{\hat{i}}, \mathbf{u}^* \rangle \end{aligned}$$

Relying on the setting of $\theta_{\hat{i}}$ being as $d_{\hat{i}}^{\mathsf{post}} - \langle \mathbf{v}_{\hat{i}}, \widetilde{\mathbf{u}} \rangle$ for $d_{\hat{i}}^{\mathsf{post}} = \langle \mathbf{v}_{\hat{i}}, \mathbf{u}^* \rangle$, the last equation thus holds.

In addition, for $\mathsf{ict}_0 = \mathbf{A}_{\text{ALS}} \cdot \mathbf{s} + \mathbf{e}_{\text{ALS},0}$, we know that $\mathsf{ReRand}(\mathbf{J}_{\hat{i}}^*, \mathsf{ict}_0, \sigma_{\text{ALS}}, \tau) = (\mathbf{B}\mathbf{J}_{\hat{i}}^*)^\top \cdot \mathbf{s} + \mathbf{e}'_{\hat{i}}$ for $\tau > s_1(\mathbf{J}_{\hat{i}}^*)$, where $\mathbf{e}'_{\hat{i}} \overset{s}{\approx} \mathcal{D}_{\mathbb{Z}^{(t+2)},2\sigma_{\text{ALS}}\tau}$ by the property of ReRand (Lemma D.1). Therefore, we obtain the following result:

$$\begin{aligned}
\beta_{1,\delta_i} &= \mathbf{D}_{\text{ALS},\delta_i} \cdot \mathbf{s} + \mathbf{e}_{\text{ALS},1,\delta_i} + p^{e-1}\mathbf{u}_{\text{ALS},b,\delta_i} + (\mathbf{B}\mathbf{J}_{\hat{i}}^*)^\top \cdot \mathbf{s} + \mathbf{e}'_{\hat{i}} + \mathbf{e}_{1,\delta_i} \\
&= \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{e}_{\text{ALS},1,\delta_i} + \mathbf{e}'_{\hat{i}} + \mathbf{e}_{1,\delta_i} + p^{e-1}\mathbf{u}_{\text{ALS},b,\delta_i} \\
&\overset{s}{\approx} \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{e}_{1,\delta_i} + p^{e-1}\mathbf{u}_{\text{ALS},b,\delta_i}.
\end{aligned}$$

The simulated transcript corresponds to $\mathcal{H}_4$ if the N-ALS challenger selects the challenge bit $b = 0$, and to $\mathcal{H}_5$ if $b = 1$. Therefore, we successfully simulate either $\mathcal{H}_4$ or $\mathcal{H}_5$ based on the challenge bit chosen by the N-ALS challenger. ◻

◻

Lemma C.17 *$\mathcal{H}_5$ and $\mathcal{H}_6$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.13 and is therefore omitted for brevity.

Lemma C.18 *$\mathcal{H}_6$ and $\mathcal{H}_7$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.14 and is therefore omitted for brevity.

Lemma C.19 *$\mathcal{H}_7$ and $\mathcal{H}_8$ are computationally indistinguishable under the* LWE *assumption.*

Lemma C.20 *$\mathcal{H}_8$ and $\mathcal{H}_9$ are computationally indistinguishable under the* LWE *assumption.*

The proof of Lemma C.19 and C.20 are similar to the proof of Lemma C.6 and are therefore omitted for brevity.

Lemma C.21 *$\mathcal{H}_9$ and $\mathcal{H}_{10}$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.16 and is therefore omitted for brevity.

Lemma C.22 *$\mathcal{H}_{10}$ and $\mathcal{H}_{11}$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.17 and is therefore omitted for brevity.

Lemma C.23 *$\mathcal{H}_{11}$ and $\mathcal{H}_{12}$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.18 and is therefore omitted for brevity.

Lemma C.24 *$\mathcal{H}_{12}$ and $\mathcal{H}_{13}$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.20 and is therefore omitted for brevity.

Lemma C.25 *$\mathcal{H}_{13}$ and $\mathcal{H}_{14}$ are statistically indistinguishable.*

The proof of this lemma is similar to the proof of Lemma B.22 and is therefore omitted for brevity. ◻

◻

D Inner Product Functional Encryption Schemes

In this section, we review IPFE schemes which are required for security proofs of our proposed predicate IPFE schemes.

D.1 ALS Inner Product Functional Encryption Scheme [ACGU20]

We adopt the ALS IPFE scheme introduced in [ACGU20], which can be proven secure under the standard LWE assumption using the noise re-randomization technique during proof. Specifically, we consider the message vector space $\mathcal{U} = \{1, \ldots, U-1\}^t$ and the key vector space $\mathcal{V} = \{1, \ldots, V-1\}^t$ for some integer U, P and dimension $t = \mathsf{poly}(\lambda)$. The inner products are evaluated over $\mathbb{Z}$ and belongs to $\{1, \ldots, Y-1\}$ with $Y = tUV$.

Construction 5 (Inner Product Functional Encryption from LWE [ACGU20]).

$\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* 1^λ*,*

1. *Set parameters* n, m, σ, ρ, q.
2. *Sample* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ *and* $\mathbf{Z} \leftarrow \mathcal{D}_{\mathbb{Z}^{t \times m}, \rho}$.
3. *Compute* $\mathbf{D} = \mathbf{Z} \cdot \mathbf{A} \in \mathbb{Z}_q^{t \times n}$.
4. *Output the public and master secret keys*

$$\mathsf{mpk} := (\mathbf{A}, \mathbf{D}), \mathsf{msk} := \mathbf{Z}.$$

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{v})$ *takes as input* msk *and key vector* $\mathbf{v}$*, compute and return* $\mathsf{sk}_{\mathbf{v}} := \mathbf{z}_{\mathbf{v}} = \mathbf{Z}^\top \mathbf{v}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{u})$ *takes as input* mpk *and a message* $\mathbf{u}$*,*

1. *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma}$, $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^t, \sigma}$.
2. *Compute*

$$\mathsf{ct}_0 := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}_0,\ \mathsf{ct}_1 := \mathbf{D} \cdot \mathbf{s} + \mathbf{e}_1 + \lfloor \tfrac{q}{Y} \rfloor \cdot \mathbf{u}.$$

3. *Output the ciphertext* $\mathsf{ct} := (\mathsf{ct}_0, \mathsf{ct}_1)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *takes as input* sk *and* ct*,*

1. *Compute* $\mu' = \mathbf{v}^\top \mathsf{ct}_1 - \mathbf{z}_{\mathbf{v}}^\top \mathsf{ct}_0 \mod q$.
2. *Output* $\mu \in \{0, \ldots, Y+1\}$ *that minimizes* $|\lfloor \frac{q}{Y} \rfloor \cdot \mu - \mu'|$.

Parameters Setting. The parameters should satisfy the following constraints aiming for the correctness and security.

1. The final magnitude of decryption error must be less than $\frac{q}{2Y}$ for the correctness.
2. To ensure the hardness of $\mathsf{LWE}_{q,n,\alpha}$, we require $\alpha q \geq \Omega(n)$.
3. We require $\tau > s_1(\mathbf{Z})$ in order to rely on ReRand algorithm (Lemma D.1) for security proof. According to Lemma A.1, $s_1(\mathbf{Z})$ is bounded by $1/\sqrt{2\pi} \cdot \rho \cdot (\sqrt{t} + \sqrt{m} + \sqrt{\lambda})$.
4. To ensure large enough entropy, we require $\rho > \omega(\sqrt{\log \lambda})$ and $m \geq (2n \log q + 2n)/\log(4/3)$.

The parameters could be chosen as: $n = \mathsf{poly}(\lambda)$, $m = 2(n \log q)$, $\rho > \omega(\sqrt{\log \lambda})$, $\tau = 1/\sqrt{2\pi} \cdot \rho \cdot (\sqrt{t} + \sqrt{m} + \sqrt{\lambda})$, $\sigma = 2\alpha q \tau$, $q > 2Yt\sqrt{t}\omega(\log^2 n)$.

Lemma D.1 (Noise Rerandomization [KY16]) *Let* q, ℓ, m *be positive integers and* r *a positive real satisfying* $r > \max\{\eta_\epsilon(\mathbb{Z}^m), \eta_\epsilon(\mathbb{Z}^\ell)\}$. *Let* $\mathbf{b} \in \mathbb{Z}_q^m$ *be arbitrary and* $\mathbf{x}$ *chosen from* $\mathcal{D}_{\mathbb{Z}^m, r}$. *Then for any* $\mathbf{V} \in \mathbb{Z}^{m \times \ell}$ *and positive real* $\sigma > s_1(\mathbf{V})$*, there exists a PPT algorithm* $\mathsf{ReRand}(\mathbf{V}, \mathbf{b} + \mathbf{x}, r, \sigma)$ *that outputs* $\mathbf{b}' = \mathbf{b}\mathbf{V} + \mathbf{x}'$ *where the statistical distance of the discrete Gaussian* $\mathcal{D}_{\mathbb{Z}^\ell, 2r\sigma}$ *and the distribution of* $\mathbf{x}'$ *is within* 8ϵ.

D.2 N-ALS Inner Product Functional Encryption Scheme [WFL19]

We now review the N-ALS IPFE scheme proposed in [WFL19] with the master secret key $\mathbf{Z}_k$ chosen from discrete Gaussian, as in [ACGU20]. Specifically, we consider the inner products modulo prime p, the plaintext and key vectors belong to $\mathbb{Z}_p^t$.

Construction 6 (N-ALS Inner Product Functional Encryption from LWE [WFL19]).

$\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* 1^λ,

1. *Set parameters* $n, m, \sigma, \rho, q = p^k$ *for some integer* k.
2. *Sample* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$ *and* $\mathbf{Z}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{t\times m},\rho}$ *for* $k \in [N]$.
3. *Compute* $\mathbf{D}_k = \mathbf{Z}_k \cdot \mathbf{A} \in \mathbb{Z}_q^{t\times n}$.
4. *Output the public and master secret keys*

$$\mathsf{mpk} := (\mathbf{A}, \{\mathbf{D}_k\}_{k\in[N]}), \mathsf{msk} := \{\mathbf{Z}_k\}_{k\in[N]}.$$

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{v})$ *takes as input* msk *and key vector* $\mathbf{v} = (\mathbf{v}_1^\top, \ldots, \mathbf{v}_N^\top) \in \mathbb{Z}_p^{N\cdot t}$, *compute and return* $\mathsf{sk}_\mathbf{v} := \sum_{k\in[N]}(\mathbf{Z}_k^\top \mathbf{v}_i)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{u})$ *takes as input* mpk *and a message* $\mathbf{u} = (\mathbf{u}_1^\top, \ldots, \mathbf{u}_N^\top) \in \mathbb{Z}_p^{N\cdot t}$,

1. *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, $\mathbf{e}_{1,k} \leftarrow \mathcal{D}_{\mathbb{Z}^t,\sigma}$.
2. *Compute*

$$\mathsf{ct}_0 := \mathbf{A}\cdot\mathbf{s} + \mathbf{e}_0,\ \mathsf{ct}_{1,k} := \mathbf{D}\cdot\mathbf{s} + \mathbf{e}_{1,k} + p^{k-1}\cdot\mathbf{u}_i \textit{ for } k \in [N].$$

3. *Output the ciphertext* $\mathsf{ct} := (\mathsf{ct}_0, \{\mathsf{ct}_{1,k}\}_{k\in[N]})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *takes as input* sk *and* ct,

1. *Compute* $\mu' = \sum_{k\in[N]} \mathbf{v}_i^\top \mathsf{ct}_{1,k} - \mathbf{z}_\mathbf{v}^\top \mathsf{ct}_0 \mod q$.
2. *Output* $\mu \in \mathbb{Z}_p$ *that minimizes* $|p^{k-1}\cdot\mu - \mu'|$.

Parameters Setting. The parameters should satisfy the following constraints aiming for the correctness and security.

1. The final magnitude of decryption error must be less than $\frac{1}{2}p^{k-1}$ for the correctness.
2. To ensure the hardness of $\mathsf{LWE}_{q,n,\alpha}$, we require $\alpha q \geq \Omega(n)$.
3. We require $\tau > s_1(\mathbf{Z}_k)$ in order to rely on ReRand algorithm (Lemma D.1) for security proof. According to Lemma A.1, $s_1(\mathbf{Z}_k)$ is bounded by $1/\sqrt{2\pi}\cdot\rho\cdot(\sqrt{t}+\sqrt{m}+\sqrt{\lambda})$.
4. To ensure large enough entropy, we require $\rho > \omega(\sqrt{\log\lambda})$ and $m \geq (2n\log q + 2n)/\log(4/3)$.

The parameters could be chosen as: $n = \mathsf{poly}(\lambda)$, $m = O(n\log q)$, $\rho > \omega(\sqrt{\log\lambda})$, $\tau = 1/\sqrt{2\pi}\cdot\rho\cdot(\sqrt{t} + \sqrt{m} + \sqrt{\lambda})$, $\sigma = 2\alpha q\tau$, $q = N\cdot p^2\cdot\sigma t(1+\rho m)$.

E Predicate Encryption with Semi-adaptive security

In this section, we present (Q, poly) semi-adaptively secure predicate encryption scheme, constructed from (Q, poly)-sel PE scheme introduced in Section 3.2 and the upgrading approach proposed in [BV16]. At a high level, our construction idea is similar to that in [LLW21], which is also inspired by [BV16].

In the selective security game of a PE scheme, the adversary is asked to submit its challenge attribute $\mathbf{x}^*$ at the very beginning. The challenger can then encode the information of $\mathbf{x}^*$ into public parameters. This preparation is commonly used for further reductions and key generations. In the semi-adaptive security game, however, the adversary is allowed to submit its challenge attribute $\mathbf{x}^*$ after the Setup phase (but before any secret key queries). Thus, the challenger in the semi-adaptive game must complete Setup without having access to the challenge attribute information.

As suggested in [BV16, LLW21], one approach is to use a substitute to play the role of the challenge attribute during the Setup process. More precisely, in the absence of information about $\mathbf{x}^*$, we can still sample a random string $\mathbf{r}$ and use it as a placeholder for the challenge attribute in generating both public parameters and the challenge ciphertext. To ensure correctness, the secret key for a predicate f is generated for an offset function $f_{\mathbf{r}'}$, defined as $f_{\mathbf{r}'}(\mathbf{x}) = f(\mathbf{x}\oplus\mathbf{r}')$, rather than for f itself. Specifically, $\mathbf{r}'$ is set as $\mathbf{r}\oplus\mathbf{x}^*$ for each secret key, once the challenge attribute $\mathbf{x}^*$ is available. According to the security definition, the challenger indeed obtains $\mathbf{x}^*$ before receiving any key queries. As a result, we have $f_{\mathbf{r}'}(\mathbf{r}) = f(\mathbf{r} \oplus \mathbf{r}^*) = f(\mathbf{r} \oplus \mathbf{r} \oplus \mathbf{x}^*) = f(\mathbf{x}^*)$, ensuring the consistency of predicate function results. Clearly, the secret random offset $\mathbf{r}$ is the key for achieving semi-adaptive security. On the other hand, in the normal scheme, to match offset functions $f_{\mathbf{r}}$ for a random $\mathbf{r}$, ciphertexts should be computed for $\mathbf{x} \oplus \mathbf{r}$ accordingly. However, we cannot publish $\mathbf{r}$ as part of public encryption key. Therefore, an encryptor is required to generate ciphertext for all possible $\mathbf{r}$ first. To prevent security leakage, these ciphertext are then encrypted by an outer-layer encryption scheme. In addition, the corresponding decryption keys for the outer-layer encryption are included in each secret key.

Technically, although we adopt a similar upgrading approach, our semi-adaptively secure bounded collusion PE scheme is more compact than the scheme in [LLW21]. The primary reason is that both the private attribute (i.e., FHE secret key) and the majority public attribute (i.e., dummy FHE ciphertexts) have been eliminated in our construction. In particular, these additional attributes needs to be encrypted for two layers: first by the underlying PE scheme and then by the outer encryption. This results in additional overhead for both ciphertexts and the corresponding public keys (inclued in the final master public key). Detail comparisons are provided in Table 2.

	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{ct}\rvert$
[LLW21]	$(O(Q) + \ell \cdot \lvert\mathsf{hct}\rvert) \cdot \lvert\mathbb{Z}_q^{n\times m}\rvert$ $+2\ell \cdot \lvert\mathsf{hct}\rvert \cdot \lvert\mathsf{pke.pk}\rvert$ $+(O(Q) \cdot \lvert\mathsf{hct}\rvert + \lvert\mathsf{hsk}\rvert) \cdot \lvert\mathbb{Z}_q^{n\times m}\rvert$ $+2\ell(O(Q) \cdot \lvert\mathsf{hct}\rvert + \lvert\mathsf{hsk}\rvert) \cdot \lvert\mathsf{pke.pk}\rvert$	$O(Q)\lvert\mathbb{Z}_q^m\rvert + 2\ell \cdot \lvert\mathsf{hct}\rvert \cdot \lvert\mathsf{pke.ct}\rvert(1 + \lvert\mathbb{Z}_q^m\rvert)$ $2O(Q)\lvert\mathsf{hct}\rvert \cdot \lvert\mathsf{pke.ct}\rvert(1 + \lvert\mathbb{Z}_q^m\rvert)$ $2\lvert\mathsf{hsk}\rvert \cdot \lvert\mathbb{Z}_q^m\rvert \cdot \lvert\mathsf{pke.ct}\rvert$
Ours	$(O(Q) + \ell \cdot \lvert\mathsf{hct}\rvert) \cdot \lvert\mathbb{Z}_q^{n\times m}\rvert$ $+2\ell \cdot \lvert\mathsf{hct}\rvert \cdot \lvert\mathsf{pke.pk}\rvert$	$O(Q)\lvert\mathbb{Z}_q^m\rvert + 2\ell \cdot \lvert\mathsf{hct}\rvert \cdot \lvert\mathsf{pke.ct}\rvert(1 + \lvert\mathbb{Z}_q^m\rvert)$

Table 2. Comparison with semi-adaptively Q-collusion resistant PE construction in [LLW21]. We denote the bit-length of attribute as ℓ, the size of homomorphic encryption ciphertext (for 1-bit) and secret key by $\lvert\mathsf{hct}\rvert$ and $\lvert\mathsf{hsk}\rvert$, respectively. We denote the size of ciphertext and public key of public key encryption scheme by $\lvert\mathsf{pke.ct}\rvert$ and $\lvert\mathsf{pke.pk}\rvert$, respectively. The size of an element in $\mathbb{Z}_q^{n\times m}$ (resp. $\mathbb{Z}_q^m$) is denoted by $\lvert\mathbb{Z}_q^{n\times m}\rvert$ (resp. $\lvert\mathbb{Z}_q^m\rvert$).

Construction 7 **((Q, poly) semi-adaptively secure PE).**

Our construction uses the following building blocks:

- *a* (Q, poly) *selectively secure PE scheme* $\mathsf{PE_{sel}} = (\mathsf{QPE.Setup}, \mathsf{QPE.KeyGen}, \mathsf{QPE.Enc}, \mathsf{QPE.Dec})$ *for predicate space* $\mathcal{F}$*, attribute space* $\mathcal{X}$ *and message space* $\mathcal{M}$*. Specifically, the encryption algorithm* $\mathsf{QPE.Enc}$ *is decomposed into two parts:* $\mathsf{QPE.Enc_{msg}}(\mu; \mathsf{rand})$ *and* $\{\mathsf{QPE.Enc_{attr}}(x_i; \mathsf{rand})\}_{i\in[\ell]}$*, where* rand *is the common randomness within two sub-algorithms.*
- *a semantically secure public key encryption scheme* $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$.

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d, 1^Q)$ *Given as input the security parameter* λ*, the attribute length* ℓ*, the depth of the circuit family* d*, and* Q *as the upper bound of 1-key queries, does the following:*

1. *Run* $(\mathsf{mpk_{sel}}, \mathsf{msk_{sel}}) \leftarrow \mathsf{QPE.Setup}(1^\lambda, 1^\ell, 1^d, 1^Q)$.
2. *Run* $\mathsf{PKE.Gen}(1^\lambda)$ *for* 2ℓ *times to get* $\{(\mathsf{PKE.pk}_{i,b}, \mathsf{PKE.sk}_{i,b})\}_{i\in[\ell],b\in\{0,1\}}$.
3. *Sample* $\mathbf{r} \xleftarrow{\$} \{0,1\}^\ell$.
4. *Output* $\mathsf{mpk} = (\mathsf{mpk_{sel}}, \{(\mathsf{PKE.pk}_{i,b})\}_{i\in[\ell],b\in\{0,1\}})$ *and* $\mathsf{msk} = (\mathsf{msk_{sel}}, \{(\mathsf{PKE.sk}_{i,b})\}_{i\in[\ell],b\in\{0,1\}}, \mathbf{r})$.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ *Given as input the master secret key* msk *and a circuit* $f \in \mathcal{F}$*, does the following:*

1. *Define a function* $f_{\mathbf{r}}(\mathbf{x}) \coloneqq f(\mathbf{x} \oplus \mathbf{r})$.

2. *Generate* $\mathsf{sk}_{\mathsf{sel},f} \leftarrow \mathsf{QPE.KeyGen}(\mathsf{msk}_{\mathsf{sel}}, f_{\mathbf{r}})$.

3. *Return* $\mathsf{sk}_f := (\mathsf{sk}_{\mathsf{sel},f}, \mathbf{r}, \{(\mathsf{PKE.sk}_{i,r_i})\}_{i\in[\ell]})$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ *Given as input the master public key, an attribute* $\mathbf{x} \in \{0,1\}^\ell$ *and a message* μ, *does the following:*

1. *Sample randomness* rand *and run*

$$\mathsf{ct}_{\mathsf{msg}} \leftarrow \mathsf{QPE.Enc}_{\mathsf{msg}}(\mu; \mathsf{rand}),$$
$$\{\mathsf{ct}_{\mathsf{attr},i,b} \leftarrow \mathsf{QPE.Enc}_{\mathsf{attr}}(x_i \oplus b; \mathsf{rand})\}_{i\in[\ell],b\in\{0,1\}}.$$

2. *Compute* $\{\mathsf{PKE.ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_{i,b}, \mathsf{ct}_{\mathsf{attr},i,b})\}_{i\in[\ell],b\in\{0,1\}}$.

3. *Return* $\mathsf{ct} = (\mathsf{ct}_{\mathsf{msg}}, \{\mathsf{PKE.ct}_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$ *Given as input a secret key and a ciphertext, does the following:*

1. *Select* $\{\mathsf{PKE.ct}_{i,r_i}\}_{i\in[\ell]}$ *according to* $\mathbf{r}$.

2. *Compute* $\mathsf{ct}_{\mathsf{attr},i,r_i} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.ct}_{i,r_i}, \mathsf{PKE.sk}_{i,r_i})$.

3. *Run* $\mathsf{QPE.Dec}$ *taking input as* $\mathsf{ct}_{\mathsf{sel}} = (\mathsf{ct}_{\mathsf{msg}}, \{\mathsf{ct}_{\mathsf{attr},i,r_i}\}_{i\in[\ell]})$ *and* $\mathsf{sk}_{\mathsf{sel},f}$.

4. *Return the output of* $\mathsf{QPE.Dec}$.

Correctness. The correctness of the construction 7 follows from the correctness of the underlying PKE schemes and the designing of each offset functions. Firstly, a bunch of PKE decryption reveals the original attribute-related ciphertext $\{\mathsf{ct}_{\mathsf{attr},i,r_i}\}_{i\in[\ell]}$ for the attribute $\mathbf{x} \oplus \mathbf{r}$ of the underlying selectively secure PE scheme. This thus allows to reconstruct a complete PE ciphertext $\mathsf{ct}_{\mathsf{sel}} = (\mathsf{ct}_{\mathsf{msg}}, \{\mathsf{ct}_{\mathsf{attr},i,r_i}\}_{i\in[\ell]})$. Secondly, by generating secret key for $f_{\mathbf{r}}$ that satisfies $f_{\mathbf{r}}(\mathbf{x} \oplus \mathbf{r}) := f(\mathbf{x} \oplus \mathbf{r} \oplus \mathbf{r}) = f(\mathbf{x})$, we can correctly decrypt the ciphertext by running the decryption of the underlying PE scheme on input $\mathsf{ct}_{\mathsf{sel}}$ and $\mathsf{sk}_{\mathsf{sel},f}$.

Security.

Theorem E.1 *Assume that* PKE *is semantically secure and* $\mathsf{PE}_{\mathsf{sel}}$ *is* (Q, poly)-sel-*SIM secure for the predicate class* $\mathcal{F}$, *then the construction 7 is* (Q, poly)-sa-*SIM secure for the same predicate class* $\mathcal{F}$, *according to Definition 2.*

The high level proof idea is similar to that of prior schemes [BV16, LLW21]. For conciseness, we outline a proof sketch below.

Proof (sketch). To construct the simulator Sim, we need to rely on the simulator $\mathsf{Sim}_{\mathsf{sel}} = (\mathsf{QSetup}^*, \mathsf{QKeyGen}^*_{\mathsf{pre}}, \mathsf{QEnc}^*, \mathsf{QKeyGen}^*_{\mathsf{post}})$.

Simulator. $\mathsf{Sim}(1^\lambda, 1^{|\mathbf{x}|})$:

1. $\mathsf{Setup}^*(1^\lambda, 1^{|\mathbf{x}|})$ generates PKE scheme key pairs and samples $\mathbf{r}$ as in the real scheme. After receiving $\mathsf{mpk}_{\mathsf{sel}}$ from the simulator QSetup^*, it sets and returns $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{sel}}, \{(\mathsf{PKE.pk}_{i,b})\}_{i\in[\ell],b\in\{0,1\}})$ Then, it initializes $\mathsf{st} := \emptyset$.
2. $\mathsf{KeyGen}^*_{\mathsf{pre}}(\mathsf{st}, f)$ first defines the function $f_{\mathbf{r}}(\mathbf{x}) := f(\mathbf{x} \oplus \mathbf{r})$ and submits key query $f_{\mathbf{r}}$ to $\mathsf{QKeyGen}^*_{\mathsf{pre}}$. Once receiving $\mathsf{sk}_{\mathsf{sel},f}$, it sets and returns $\mathsf{sk}_f = (\mathsf{sk}_{\mathsf{sel},f}, \mathbf{r}, \{(\mathsf{PKE.sk}_{i,r_i})\}_{i\in[\ell]})$, as in the real scheme. Additionally, it updates st with 1-key queries information.
3. $\mathsf{Enc}^*(\mathsf{st})$ first invokes QEnc^* with the input st. After receiving the challenge ciphertext $\mathsf{ct}^*_{\mathsf{sel}}$, it parses $\mathsf{ct}^*_{\mathsf{sel}}$ as $(\mathsf{ct}_{\mathsf{sel},\mathsf{msg}}, \{\mathsf{ct}_{\mathsf{sel},\mathsf{attr},i}\}_{i\in[\ell]})$. Next, it sets $\{\mathsf{ct}_{\mathsf{sel},\mathsf{attr},i,1-r_i}\}_{i\in[\ell]}$ as uniformly random from corresponding ciphertext space. Then, it computes $\{\mathsf{PKE.ct}_{i,1-r_i} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_{i,1-r_i}, \mathsf{ct}_{\mathsf{sel},\mathsf{attr},i,1-r_i})\}_{i\in[\ell]}$ and $\{\mathsf{PKE.ct}_{i,r_i} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_{i,r_i}, \mathsf{ct}_{\mathsf{sel},\mathsf{attr},i})\}_{i\in[\ell]}$ as in the real scheme. Finally, it returns $\mathsf{ct} = (\mathsf{ct}_{\mathsf{msg}}, \{\mathsf{PKE.ct}_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$.
4. $\mathsf{QKeyGen}^*_{\mathsf{post}}(\mathsf{st}, f)$ generates secret keys and maintains st as in the $\mathsf{KeyGen}^*_{\mathsf{pre}}$.

Starting from the real experiment of semi-adaptive security (denoted by $\mathcal{H}_0$), we can first replace the ciphertext components $\{\mathsf{PKE.ct}_{i,1-r_i}\}_{i\in[\ell]}$ with the PKE encryptions of random plaintexts. $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$ then follows from the semantically security of underlying PKE schemes. Clearly, the distinguishability advantage between $\mathcal{H}_1$ and the ideal experiment is bounded by the security of selectively secure PE scheme. For a detailed proof, we refer the reader to [BV16]. □

□

Furthermore, by applying the similar transformation, our proposed P-IPFE scheme can also be upgraded to semi-adaptively secure.

Corollary 1. *Assuming the hardness of* LWE *with appropriate parameter choices, there exists a predicate IPFE scheme with* (Q, poly)-sa-*SIM security.*