New Lattice Two-Stage Sampling Technique and its Applications to Functional Encryption – Stronger Security and Smaller Ciphertexts

Qiqi Lai[1,2], Feng-Hao Liu[3], Zhedong Wang[4] (Corresponding Author)

[1] School of Computer Science, Shaanxi Normal University, Xi'an, China
laiqq@snnu.edu.cn.
[2] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China
[3] Florida Atlantic University, Boca Raton, FL, USA
fenghao.liu@fau.edu.
[4] School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
wzdstill@sjtu.edu.cn.

Abstract. This work proposes a new two-stage lattice two-stage sampling technique, generalizing the prior two-stage sampling method of Gentry, Peikert, and Vaikuntanathan (STOC '08). By using our new technique as a key building block, we can significantly improve security and efficiency of the current state of the arts of simulation-based functional encryption. Particularly, our functional encryption achieves (Q, poly) simulation-based semi-adaptive security that allows arbitrary pre- and post-challenge key queries, and has succinct ciphertexts with only an additive $O(Q)$ overhead.
Additionally, our two-stage sampling technique can derive new feasibilities of indistinguishability-based adaptively-secure IB-FE for inner products and semi-adaptively-secure AB-FE for inner products, breaking several technical limitations of the recent work by Abdalla, Catalano, Gay, and Ursu (Asiacrypt '20).

1 Introduction

Functional Encryption (FE) [14, 43] is a powerful generalization of public-key encryption (PKE), allowing more fine-grained information disclosure to a secret key holder. FE with regular syntax can be described as follows – every secret key is associated with a function f (in some class $\mathcal{F}$), and the decryptor given such key (i.e., sk_f) and a ciphertext $\mathsf{Enc}(u)$ can only learn $f(u)$. During the past decade, there has been tremendous progress of FE for various function classes, e.g., [2, 4–6, 26, 32, 33] and more.

To facilitate presentation and comparisons with prior work, we consider the notion of FE with a more fine-grained syntax, which has been studied in the literature to capture various settings of FE [1,2,14,33]. Particularly, each message

u consists of two parts, namely $u := (\mathsf{x}, \mu)$, where x is some index (or attribute)[5], and μ is some message. Additionally, each function f consists of two parts, namely, $f := (\mathsf{P}, g) \in \mathcal{P} \times \mathcal{G}$, where P is a predicate over the index, and g is a function over the message. The overall function acts as:

$$f(u) := \begin{cases} g(\mu) & \text{if } \mathsf{P}(\mathsf{x}) = 1 \\ \perp & \text{otherwise.} \end{cases}$$

When decrypting the ciphertext $\mathsf{ct}_u = \mathsf{Enc}(\mathsf{x}, \mu)$ by $\mathsf{sk}_f := \mathsf{sk}_{(\mathsf{P},g)}$, the decryptor can learn $g(\mu)$ if $\mathsf{P}(\mathsf{x}) = 1$, and $\perp$ otherwise. Under this syntax, we call a key $\mathsf{sk}_{f:=(\mathsf{P},g)}$ a 1-key with respect to an index x if $\mathsf{P}(\mathsf{x}) = 1$, or otherwise a 0-key. Intuitively, a 1-key is allowed to open the ciphertext, but a 0-key is not.

Even though FE with the fine-grained syntax is essentially equivalent to the regular syntax for sufficiently expressive function/predicate classes, it is more convenient to present our new results in this way. Moreover as noticed since [14], many advanced encryption schemes such as identity-based encryption, attribute-based encryption, predicate encryption can be captured naturally from this notion, by different predicate and function classes $\mathcal{P} \times \mathcal{G}$.

There are two important settings studied in the literature – FE with private or public index, according to whether the index x is revealed to the decryption algorithm. In what follows, we first discuss in more details about challenges of the state of the arts in both settings. Then we present our contributions and new techniques to break these barriers and advance the research frontiers.

FE with Private Index. In this setting, FE provides very strong privacy guarantee where only $g(\mu)$ can be learned given a 1-key $\mathsf{sk}_{\mathsf{P},g}$ and a $\mathsf{Enc}(\mathsf{x}, \mu)$ with $\mathsf{P}(\mathsf{x}) = 1$. It is worthwhile to point out that in this setting, realizing the class $\mathcal{P} \times \{I\}$ for the identity function I is already general enough, as it suffices to capture FE (of regular syntax) for the boolean circuit class $\mathcal{P}$. In particular, we can use $\mathsf{sk}_{\mathsf{P},I}$ and $\mathsf{Enc}(\mathsf{x}, \mu)$ to simulate the exact effect of sk_{P} and $\mathsf{Enc}(\mathsf{x})$ of the regular syntax FE. Therefore, following some prior work [2], this work just focuses on the function class $\mathcal{P} \times \{I\}$ for FE in the private index setting by default. We discuss this in more details in Section 3.1.

To capture security, there have been notions of indistinguishable-based (IND) and simulation-based (SIM) definitions proposed and studied in the literature since [14]. As raised by [14], the IND-based security is inadequate (i.e., too weak) in the private index setting for certain functionalities and applications. Thus, it would be much desirable to achieve the stronger notion of SIM-based notion.

However, there are various settings that the SIM-based notion is too strong to be attained. For example, the work [14] showed that for very simple functionalities (identity-based encryption), the SIM-based security is impossible for multiple challenge ciphertexts, even given just one post-challenge key query. Additionally, the work [4] showed that for FE scheme with respect to general function class, the ciphertext size must grow linearly with the number of pre-challenge key

[5] We note that both the names "index" and "attribute" have been used interchangeably in the literature.

queries. Therefore it is impossible to achieve the notion (poly, poly)-SIM security (allowing an unbounded number of both 1 and 0-keys) for general functions.

Despite these lower bounds, the work [33] identified important feasible settings for SIM-based security, by proposing new constructions in the setting of single challenge ciphertext and bounded collusion. More specifically, [33] achieved (Q, Q)-adaptive-SIM FE for the family of polynomial-sized circuits under the minimal assumption of PKE. Their attained SIM notion is very strong – the challenge index can be adaptively chosen and the adversary is allowed to query both pre- and post-challenge key queries, up to some bounded Q times for both 1 and 0-keys. The ciphertexts however, are not succinct (i.e., dependent on the circuit description), and their size grows with a multiplicative factor of $O(Q^4)$. Even though a recent work [11] improved the multiplicative factor to $O(Q)$ and proved the security of SIM FE solely on PKE, their ciphertexts are still not succinct. The non-succinctness of the ciphertext prohibits other important applications, such as reusable garbled circuits [32], and thus improving in this dimension would be very significant.

A subsequent work [32] constructed the first single-key *succinct* FE for bounded depth circuits, and showed that this suffices for reusable garbled circuits, solving a long-term open question in the field. However, their scheme [32] has drawbacks in the following two aspects. First, the single-key FE of [32] achieves a weaker notion of selective security and only allows one pre-challenge key query (either a 1 or 0-key). Second, even though the single-key FE of [32] can be bootstrapped to Q key FE using the compiler of [33], yet the resulting ciphertexts grows with $O(Q^4)$ multiplicatively.

Tackling these drawbacks, two almost concurrent work [2, 6] advanced this direction of work significantly. Particularly, the work [6] constructed a single key succinct FE for NC1, and then showed another bootstrapping method (from NC1 to general circuits) that only occurs an $O(Q^2)$ additive overhead, yet the resulting (offline-part) ciphertexts become no longer succinct. The other concurrent work [2] designed a new succinct single key FE that supports $(1, \mathsf{poly})$ queries for general circuits, and a new bootstrapping method that achieves (Q, poly)-SIM security with succinct ciphertexts and $O(Q^2)$ additive overhead. As a substantial milestone, [2] for the first time identified an important and useful[6] subclass of key queries (i.e., 0-keys), where SIM-based security is feasible beyond bounded collusion. Recently, the work [11] designed a simple yet very novel compiler that turns any bounded-collusion FE into one with ciphertext growth $O(Q)$ multiplicatively. This compiler improves the ciphertext size significantly, but does not improve the security over the original scheme.

Challenges. The attainable SIM-based security of [2] is however weaker than that of the work [33] in three aspects – (1) the challenge index needs to be semi-adaptive (the adversary commits to the challenge right after the master public-key); (2) the 1-key queries need to be made at one-shot right before the

[6] For example in IBE and ABE, 0-keys are useful for decrypting other ciphertexts with satisfying indices. They just cannot decrypt the specific (challenge) index.

challenge ciphertext; (3) no more 1-key is allowed for post-challenge phase. How to bridge the gap between the two methods [2,33] is an important open question.

To measure how large the gap is, we first notice that the semi-adaptive attribute (i.e., aspect (1)) can be mitigated (though not completely satisfactory) by the generic complexity leveraging argument as also pointed out by [32]. Particularly, by scaling up the ℓ in bit-security of the selective scheme, we can achieve adaptive security over ℓ-bit index. Even though theoretically this would require to assume sub-exponential security of the underlying hard problem, yet nevertheless in practice this assumption is usually in use, given the estimations of the best-known concrete attacks, e.g., the concrete LWE estimation [7].

On the other hand, how to tackle adaptiveness for pre-challenge and post-challenge key queries seems beyond the current techniques, as the length to describe all possible key queries requires $Q \cdot \mathsf{poly}(\lambda)$ bits for some unbounded polynomial, which is too large for the complexity leveraging argument. Thus, how to improve aspects (2) and (3) would require substantial new techniques. This work aims to solve these challenges with the following particular goal.

> (**Main Goal 1:**) Design a succinct FE for general bounded depth circuits with (Q, poly)-SIM-based security[7], allowing arbitrary pre- and post-challenge queries for both 1 and 0-keys.

FE with Public Index. The public index setting does not require the scheme to hide the index, and for many scenarios in this setting the IND security notion would already be adequate, as pointed out by [14]. Even though FE with public index can be generically derived from FE with private index, much more efficient solutions are desired. For example, current instantiations of FE with private index either use heavy tools such as garbled circuits or fully homomorphic encryption, while the identity-based encryption [3] (as a special case of FE with public index) only requires simple lattice operations and thus can be much more efficient.

A recent work [1] studied the class $\mathsf{IB} \times \mathsf{IP}$, where the IB is the class of identity comparison predicates and IP is the class of inner products. Particularly, the work [1] showed that by connecting ABB [3] encoding for IB and ALS [5] encoding for IP, one can derive a simple FE for $\mathsf{IB} \times \mathsf{IP}$ from lattices. Albeit simple and efficient, the work [1] can only prove the selective security (over IB) for their lattice design in the standard model, even though the ABB and ALS encodings both achieve the adaptive security in their encryption settings. Moreover, note that their construction idea [1] naturally extends to the setting of $\mathsf{AB} \times \mathsf{IP}$ by connecting the AB encoding of [12] with ALS, where AB is the general attribute-based policy functions. However, their proof of security [1] even for the selective security would hit a subtle yet challenging technical barrier. Our second goal is to tackle these challenges.

> (**Main Goal 2:**) Determine new proof strategy for the class of $\mathsf{IB} \times \mathsf{IP}$ and $\mathsf{AB} \times \mathsf{IP}$ in the public index setting.

[7] We notice that $(\mathsf{poly}, \mathsf{poly})$ SIM-based security is not possible by the lower bound of [4]. Thus, (Q, poly) SIM-based security is the best we can hope for in this model.

1.1 Our Contributions

This work aims at the two main goals and makes three major contributions.

Contribution 1. First we propose a new lattice two-stage sampling technique, generalizing the prior GPV type two-stage sampling [29]. Using this new sampling technique, we design a unified framework that handles major challenges in our two (seemingly different) main goals as we elaborate next. The crux of our design relies on adding smudging noise over secret keys, which is critical in the analysis and conceptually new, as all prior work (to our knowledge) only considered adding smudging noise over ciphertexts, e.g., [2].

Contribution 2. By using our new pre-sampler, we improve the prior designs of [2] substantially as we elaborate below.

- Our first step is to achieve a $(1, \mathsf{poly})$ selectively secure (over the challenge index) *partially hiding predicate encryption* (PHPE), allowing general pre-challenge but no post-challenge key queries. Technically, our construction simply replaces the key generation algorithm in the very-selective PHPE of [2][8] by our new sampler. Our result at this step is already stronger than the work [2] in the following ways.
  1. We notice that our PHPE can achieve adaptive security by the complexity leveraging argument directly, yet the very-selective PHPE of [2] cannot, as the description of the function for key queries is too large.
  2. The two schemes can be upgraded to semi-adaptive security over the challenging index without the complexity leveraging, yet the transformation for ours is much more efficient. Particularly, our upgrade only applies the very light-weight method of [21,37], whereas the very-selective PHPE of [2] requires to compose PHPE with another FE (ALS [5]). Moreover, our resulting scheme allows arbitrary pre-challenge key queries, whereas the resulting scheme of [2] still requires the adversary to commit to the 1-key query before making further 0-key queries.
- Our $(1, \mathsf{poly})$ PHPE can be turned into FE by using the modified transformation of [35],[9] resulting in a *succinct* single key $(1, \mathsf{poly})$ FE that allows arbitrary pre-challenge key queries as long as there is at most one 1-key. This suffices to construct the reusable garbled circuits [32]. We present a comparison of our single key succinct FE with prior work in Table 1.
- Our next step is to achieve a succinct (Q, poly) FE that allows arbitrary pre- and post-challenge queries. To achieve this, we slightly modify the transformation (from $(1, \mathsf{poly})$ PHPE to (Q, poly) PHPE) of [2] by using the technique of secret sharing and a new way of generating cover-free sets inspired

[8] A very-selective scheme requires the adversary to commit to both the challenge index and function in the very beginning of the security experiment.

[9] In fact, the transformation of [35] can not be directly applied in our case, due to the fact that we need to apply a noise smudging technique to the used FHE noise. However, we can just insert a bootstrapping step to reduce the FHE noise into a polynomial bound again.

by [11]. By applying our new transformation to our $(1, \mathsf{poly})$ PHPE, we derive a (Q, poly) PHPE that allows arbitrary pre- and post-key queries. Then the desired FE again follows from the modified transformation of [35].

Importantly, our transformation inherits many nice properties in [2], e.g., the succinctness of the ciphertexts is preserved. Thus, our resulting FE has succinct ciphertexts, whose size grows *additively* with $O(Q)$, and are independent of the function/circuit size. Our result is better than the transformation of [11], which incurs a *multiplicative* $O(Q)$ blowup in the ciphertexts.

Ref.	(1-key,0-key)	(Pre,Post)-Challenge	Index	Succinct ct
[33]	$(a, b) : a + b = 1$	(✓, ✓)	AD	✗
[32]	$(a, b) : a + b = 1$	(✓, ✗)	$\mathsf{Sel}^\dagger$	✓
[6]	$(a, b) : a + b = 1$	(✓, ✗)	AD	✓ for NC1
[2]	$(1, \mathsf{poly})$	(✗, ✗)*	$\mathsf{SA}^\dagger$	✓
Ours	$(1, \mathsf{poly})$	(✓, ✗)	$\mathsf{SA}^\dagger$	✓

Table 1. Comparison of Prior Work of Single Key SIM-Secure Public-Key FE.

($*$) The scheme requires the adversary to commit to the 1-key query right after seeing the master public key. Then the adversary is allowed to make further arbitrary 0-key queries in the pre- and post-challenge phases, but not any more 1-key query.

(†) The selective (Sel)/semi-adaptive (SA) security can be raised to adaptive security (AD) by the complexity leveraging argument, at the cost of scaling up the security parameters.

In summary, we achieve our *Main Goal* 1 for semi-adaptive security over the challenge index, and the full-fledged of the goal if we further apply the complexity leveraging argument. Additionally, our scheme for the first time achieves succinct ciphertexts with only $O(Q)$ additive overhead. We present a comparison of our (Q, poly) FE with prior work in Table 2.

Ref.	(1-key,0-key)	(Pre,Post)-Challenge	Index	Succinct ct	Ciphertext size
[33]	(Q, Q)	(✓, ✓)	AD	✗	$\times\, O(Q^4)$
[32]+ [33]	(Q, Q)	(✓, ✗)	$\mathsf{Sel}^\dagger$	✓	$\times\, O(Q^4)$
[6]	(Q, Q)	(✓, ✗)	AD	✓ for NC1	$+\, O(Q^2)$
[2]	(Q, poly)	(✗, ✗)*	$\mathsf{SA}^\dagger$	✓	$+\, O(Q^2)$
[2]+ [11]‡	(Q, poly)	(✗, ✗)*	$\mathsf{SA}^\dagger$	✓	$\times\, O(Q)$
Ours	(Q, poly)	(✓, ✓)	$\mathsf{SA}^\dagger$	✓	$+\, O(Q)$

Table 2. Comparison of Other Private Index SIM-secure Public-Key FE.

($*$) The scheme requires the adversary to commit to all the Q 1-key queries (in one shot) right after seeing the master public key. Then the adversary is allowed to make further arbitrary 0-key queries in the pre- and post-challenge phases, but not any more 1-key query.
($\dagger$) Similar to Table 1.
($\ddagger$) The generic method in [11] can transform any bounded collusion FE scheme into one whose ciphertext size grows with $O(Q)$ multiplicatively.

Contribution 3. Finally, we identify that our new sampling technique is the key to break the technical barriers of the lattice-based analysis of [1]. Particularly, for the setting of public index, we construct new FE schemes for $\mathsf{IB} \times \mathsf{IP}$ and $\mathsf{AB} \times \mathsf{IP}$. The crux is to replace the key generation algorithm of [1] by our new pre-sampler. The novelty of this contribution majorly comes from the proof techniques. In Table 3 we compare our schemes with [1].

Reference	IB-FEIP	AB-FEIP
[1]	$(1, \mathsf{poly})$-Sel	✗
Ours	$(1, \mathsf{poly})$-AD	(Q, poly)-SA

Table 3. Comparison of Public Index IND-Based Construction.

1.2 Technical Overview

We present an overview of our new techniques. We first describe our central technique – a new two-stage sampling method, and then show how it can be used to achieve our main goals, together with further new insights. Our two-stage sampling method can be understood without the context of FE, and might find other applications. Thus we believe that this technique can be of general interests.

Two-stage Sampling Method. At a high level, we would like to sample the following two-stage distribution:

– In the first stage, a random matrix $\mathbf{A}$ and a random vector $\boldsymbol{u}$ are sampled;
– In the second stage, an arbitrary small-norm matrix $\mathbf{R}$ is first specified, and then a short vector $\boldsymbol{y}$ is sampled conditioned on $[\mathbf{A}|\mathbf{AR}]\boldsymbol{y} = \boldsymbol{u}$.
– The overall distribution consists of $(\mathbf{A}, \mathbf{AR}, \boldsymbol{u}, \boldsymbol{y})$.

In a series of lattice-based work [1–3,12,15,29,34,35], the proof framework requires to sample this distribution (or its slight variations) in two ways – with $\mathbf{A}$'s trapdoor and without $\mathbf{A}$'s trapdoor. On the one hand, given the trapdoor of $\mathbf{A}$, one can efficiently sample this distribution. On the other hand, without the trapdoor of $\mathbf{A}$, one can also sample the distribution through using $\mathbf{G}$-trapdoor [41]. In particular, if we have the $\mathbf{G}$ matrix [41] on the second part, i.e., the matrix is of the form $[\mathbf{A}|\mathbf{AR} + \gamma \cdot \mathbf{G}]$ with $\gamma \neq 0$, then this sampling task can be solved easily by the sample-right technique [3, 41]. However, our task (and the security proofs in this work) does not have the matrix $\mathbf{G}$ in the second matrix, and thus the prior technique cannot be applied to sample the required distribution.

Is this task even doable? To answer this question, we first consider a simpler case where there is no $\mathbf{R}$. Then we notice that this task is achievable via the classic GPV two-stage sampling technique: we first pre-sample $\boldsymbol{y}$, and set $\boldsymbol{u} = \mathbf{A}\boldsymbol{y}$. By setting parameters appropriately, the work [29] showed that the distributions $(\mathbf{A}, \boldsymbol{u}, \boldsymbol{y})$ generated in the two ways (with trapdoor and without trapdoor) are statistically indistinguishable. Moreover, this idea can be generalized to achieve a weaker version of our task where $\mathbf{R}$ is given in the first stage – we simply pre-sample $\boldsymbol{y}$, set $\boldsymbol{u} = [\mathbf{A}|\mathbf{AR}]\boldsymbol{y}$, and output $(\mathbf{A}, \mathbf{AR}, \boldsymbol{u}, \boldsymbol{y})$. In fact, this approach has been explored by prior work [2] in the context of functional encryption (more precisely PHPE). Due to the technical barrier that $\mathbf{R}$ must be given in the first stage, schemes using this approach achieve a weak notion of very selective PHPE, where the adversary needs to commit to the challenge index and 1-key query at the beginning. We will elaborate more on the connection of FE and PHPE later.

As we discuss above, the challenge comes from the fact that if $\mathbf{R}$ is only given in the second stage, the two-stage sampling method cannot generate $\boldsymbol{u}$ in a way that depends on $\mathbf{R}$. To tackle this, we aim to "eliminate" the effect of this matrix $\mathbf{R}$ in the two-stage sampling process. In particular we observe that if the matrix $\mathbf{R}$ has a small norm, we can "smudged" its effect by using a distribution with some larger parameter. With this intention in mind, we propose the following new two-stage sampling method:

– In the first stage, generate a random $\mathbf{A}$, and *pre-sample* $\boldsymbol{x}$ from a discrete Gaussian for some larger parameter ρ. Set $\boldsymbol{u} = \mathbf{A}\boldsymbol{x}$.
– In the second stage when $\mathbf{R}$ is given, sample $\boldsymbol{z}$ from a discrete Gaussian with a smaller parameter s, and then output $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} - \mathbf{R}\boldsymbol{z} \\ \boldsymbol{z} \end{pmatrix}$.
– The sampler outputs $(\mathbf{A}, \mathbf{AR}, \boldsymbol{u}, \boldsymbol{y})$ at the end.

Clearly the output $\boldsymbol{y}$ satisfies the equation $[\mathbf{A}|\mathbf{AR}]\boldsymbol{y} = \boldsymbol{u}$. If $\rho \gg s\|\mathbf{R}\|$, then we can intuitively think that $\boldsymbol{x}$ smudges $\mathbf{R}\boldsymbol{z}$, so $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} - \mathbf{R}\boldsymbol{z} \\ \boldsymbol{z} \end{pmatrix}$ behaves

like $\boldsymbol{y}' = \begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{z} \end{pmatrix}$ such that $[\mathbf{A}|\mathbf{AR}]\boldsymbol{y}' = \boldsymbol{u}$. By formalizing this idea, this task is achieved.

Improving FE with Private Index. Our two-stage sampling method can significantly improve FE with private index of [2]. Before presenting our insights, we first briefly review the framework of [2].

At a high level, [2] constructed FE in the following steps:

(1a) Construct a $(1, \mathsf{poly})$ very-selective partially hiding predicate encryption (PHPE) where the adversary needs to commit to the challenge index and 1-key query at the beginning of the security experiment.
(1b) Upgrade the basic scheme to $(1, \mathsf{poly})$ semi-adaptive PHPE by composing the basic scheme with ALS-FE for inner products [5].
(2) Upgrade the $(1, \mathsf{poly})$ semi-adaptive PHPE to (Q, poly) semi-adaptive PHPE. Here the transformation preserve succinct ciphertexts and only incurs an additive blow up of $O(Q^2)$.
(3) Transform the (Q, poly) semi-adaptive PHPE to (Q, poly) semi-adaptive FE. This step follows almost from [35] and an additional technique of adding smudging noise over the ciphertexts.

We notice that Step (3) is generic, so it suffices to focus on improving PHPE in Steps (1a) - (2). To facilitate presentation of our new ideas, we next identify the following four limitations in the current framework.

- First, Steps (1a) and (1b) require the adversary to commit to his 1-key challenge query before asking further 0-key queries.
- Second, the step (1b) uses a composition of FE over another FE, which could be overly complicated and inefficient.
- Third, Step (2) does not support post-challenge 1-key queries.
- Fourth, Step (2) incurs an additive overhead of $O(Q^2)$, which is incomparable with the multiplicative $O(Q)$ overhead the recent work by [11].

Next, we present our new insights to break all these limitations! To describe how our techniques work, we start with a highly simplified description of the very selective PHPE of [2]: the master public key contains matrices $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell$ for ℓ being the length of the index (private and public combined), and a matrix $\mathbf{P}$. Given a key query f, the key generation algorithm defines another related function C_f and computes $\mathbf{B}_{C_f}$ from $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ by the technique of key homomorphic evaluation [12]. Then the key generation algorithm samples $\mathsf{sk}_f := \mathbf{Y}$ such that $[\mathbf{A}|\mathbf{B}_{C_f}] \cdot \mathbf{Y} = \mathbf{P}$. Clearly, this sampling task can be easily performed if the trapdoor of $\mathbf{A}$ is given.

In the proof of security, the trapdoor of $\mathbf{A}$ is not given. Yet we can set $\mathbf{B}_i := \mathbf{A} \cdot \mathbf{R}_i + \mathsf{x}_i^* \mathbf{G}$ for challenge index $\mathbf{x}^* = (\mathsf{x}_1^*, \ldots, \mathsf{x}_\ell^*)$. (Note that here we do not need to distinguish public/private index to demonstrate our idea.) Then by the key homomorphic evaluation method, we have $[\mathbf{A}|\mathbf{B}_{C_f}] = [\mathbf{A}|\mathbf{AR}_{C_f} + C_f(\mathbf{x}^*)\mathbf{G}]$.

From the design of C_f, we have $C_f(\mathbf{x}^*) = 0$ if the key query f corresponds to a 1-key with respect to $\mathbf{x}^*$, or otherwise $C_f(\mathbf{x}^*) \neq 0$ if the key query corresponds to a 0-key. Therefore in the security analysis, one can clearly answer any 0-key queries as the $\mathbf{G}$-trapdoor appears in the second matrix.

At this moment, the reader can already see that answering the 1-key query corresponds to the two-stage sampling as we describe above. In fact, the reason why [2] starts with the very selective notion comes from the fact that the prior technique requires $\mathbf{R}_{C_f}$ to be given in the first stage. This requires the adversary to commit to the challenge 1-key function f and the challenge index at the beginning of the security experiment.

Note that by using our new two-stage sampling method for the key generation algorithm, we are able to answer the 1-key query at any moment just before the challenge ciphertext. Therefore, we can achieve $(1, \mathsf{poly})$ selective FE, allowing arbitrary pre-challenge key queries. Moreover by the very light-weight method of [21, 37], the FE can be upgraded to semi-adaptive security[10]. This solves the first two limitations, giving an improved way to achieve (1a) + (1b) of [2].

To further break the third and fourth limitations, we first briefly overview the transformation in Step (2) of [2]. At a high level, besides $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell$, the method generate additional matrices $\mathbf{P}_1, \ldots, \mathbf{P}_N$. The key generation would choose a small subset $\Delta \subseteq [N]$ of some fixed cardinality and generate $\mathsf{sk}_f := \mathbf{Y}$ such that $[\mathbf{A}|\mathbf{B}_{C_f}] \cdot \mathbf{Y} = \mathbf{P}_\Delta$, where $\mathbf{P}_\Delta = \sum_{i \in \Delta} \mathbf{P}_i$. To encrypt a message μ, the encryption algorithm just additionally generates $\boldsymbol{\beta}_{1,i} = \boldsymbol{s}^\top \mathbf{P}_i + \boldsymbol{e} + \frac{q}{2|\Delta|}\mu$ for all $i \in [N]$. The decryption algorithm can figure out $\boldsymbol{\beta}_{1,\Delta} = \sum_{i \in \Delta} \boldsymbol{\beta}_{1,i} = \boldsymbol{s}^\top \mathbf{P}_i + \boldsymbol{e}' + \frac{q}{2}\mu$, and the rest of the procedure is similar to the $(1, \mathsf{poly})$-PHPE. The work [2] requires that for Q randomly sampled sets $\Delta_1, \ldots, \Delta_Q$ in $[N]$, it is overwhelming that the sets are cover-free. By using the result of [33], this would require $N = O(Q^2)$. This explains why the transformation incurs an additive $O(Q^2)$ overhead.

To further reduce the parameter N, it suffices to generate cover-free sets more efficiently. We then construct a simple set sampler that only requires requires $N = O(Q)$, inspired by an implicit construction in the work [11]. We identify that this more efficient cover-freeness suffices for the rest of the proof.

Finally, we take care of the post-challenge key queries if the message space is small, e.g., bit encryption. (Here we do not need to place a constraint on the index length.) Our idea is to use secret sharing over the plaintext $\mu \in \{0, 1\}$, i.e., sharing $\frac{q}{2}\mu$ into $\mu_1, \ldots, \mu_N$, such that any subset Δ with some fixed cardinality would recover μ, i.e., $\mu = \sum_{i \in \Delta} \mu_i$. Then we generate ciphertexts $\boldsymbol{\beta}_{1,i} = \boldsymbol{s}^\top \mathbf{P}_i + \boldsymbol{e} + \mu_i$ for all $i \in [N]$. As a critical security proof step, we show that given all secret keys of the form $(\Delta, \mathbf{Y})$, one can only learn $\sum_{i \in \Delta} \mu_i = \mu$ but nothing more. By using this fact, we can design a ciphertext simulator, who generates simulated shares $\mu_1, \ldots, \mu_N$ and $2Q$ sets $\Delta_1, \ldots, \Delta_Q$, $\Delta'_1, \ldots, \Delta'_Q$ such that $\sum_{\Delta_i} \mu_i = q/2$,

[10] The reason why [2] cannot apply the light-weight method is because its basic construction only achieves very selective security, whereas the technique of [21, 37] can be applied to the selective security only over index.

and $\sum_{\Delta'_i} \mu_i = 0$. In this way, the post-challenge simulator can answer a 1-key query by using either $\{\Delta_i\}$ or $\{\Delta'_i\}$ according to whether $\mu = 1$ or $\mu = 0$.

Notice that the core and useful properties of the above process are that: (1) the simulation of the ciphertext does not depend on the plaintext μ; (2) the post-challenge key simulation can consistently generate a key that opens the simulated ciphertext to either $\mu = 1$ or $\mu = 0$. By further taking fine care of the details, we are able to achieve (Q, poly)-PHPE that supports arbitrary key queries and has succinct ciphertext that grows additively with $O(Q)$. This solves the third and fourth challenges as above and improves Step (2) of [2]. Clearly, this PHPE can also be transformed into an FE, following Step (3).

Improving FE with Public Index. Interestingly, the lattice-based construction of FE with public index [1] faces exactly the same technical challenge as the very selective PHPE of [2]. Our new two-stage sampling method is the key missing link of [1] to achieve adaptive $\mathsf{IB} \times \mathsf{IP}$ and semi-adaptive $\mathsf{AB} \times \mathsf{IP}$. We further elaborate on this setting in Section 7. The reader would immediately see the point even just with a glance at the construction.

1.3 Other Related Work

We notice that FE can be obtained from indistinguishable obfuscation ($i\mathcal{O}$) [26], achieving the notion of $(\mathsf{poly}, \mathsf{poly})$-$\mathsf{IND}$ adaptive security via [10]. Even though recently there has been substantial progress for instantiating $i\mathcal{O}$ [16, 27, 38], the derived FE (as is) cannot achieve the simulation-based security. This is because the $i\mathcal{O}$-based FE has ciphertext length independent of the number of collusion Q, and thus according to the lower bound of [4], the scheme cannot be SIM secure. Moreover as mentioned in [14, 33], IND-based FE does not imply SIM-based FE. Therefore for the direction of SIM-based FE, our work would shed light on new methods and feasibilities beyond what can be implied from the recent progress on the direction of $i\mathcal{O}$ [16, 27, 38].

In [22], Canetti and Chen show that single key SIM-secure private-key FE suffices to construct reusable garbled circuits. Compared with the reusable garbled circuits following naturally from our (Q, poly)-SA-SIM FE with $Q = 1$,[11] the construction in [22] achieves the stronger adaptive security with respect to index without the complexity leveraging argument, yet can only support either pre- or post-challenge key query and $\mathsf{NC1}$ circuits, rather than general circuits.

2 Preliminaries

2.1 Notations

In this paper, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of natural numbers, integers and real numbers, respectively. We use λ to denote the security parameter, which is the

[11] Notice that the reusable garbled circuits following from our SIM-secure FE can achieve SA-SIM security, and support general circuits and arbitrary pre- and post-challenge key query, even it just allow to be queried one time by its definition.

implicit input for all algorithms presented in this paper. A function $f(\lambda) > 0$ is negligible and denoted by $\mathsf{negl}(\lambda)$ if for any $c > 0$ and sufficiently large λ, $f(\lambda) < 1/\lambda^c$. A probability is called to be overwhelming if it is $1 - \mathsf{negl}(\lambda)$. A column vector is denoted by a bold lower case letter (e.g., $\boldsymbol{x}$). A matrix is denoted by a bold upper case letter (e.g., $\mathbf{A}$), and its transposition is denoted by $\mathbf{A}^\top$.

For a set D, we denote by $u \xleftarrow{\$} D$ the operation of sampling a uniformly random element u from D, and represent $|u|$ as the bit length of u. For an integer $\ell \in \mathbb{N}$, we use U_ℓ to denote the uniform distribution over $\{0,1\}^\ell$. Given a randomized algorithm or function $f(\cdot)$, we use $y \leftarrow f(x)$ to denote y as the output of f and x as input. For a distribution X, we denote by $x \leftarrow X$ the operation of sampling a random x according to the distribution X. Given two different distributions X and Y over a countable domain D, we can define their statistical distance to be $\mathsf{SD}(X,Y) = \frac{1}{2}\sum_{d\in D}|X(d) - Y(d)|$, and say that X and Y are $\mathsf{SD}(X,Y)$ close. Moreover, if $\mathsf{SD}(X,Y)$ is negligible in λ, we say that the two distributions are statistically close, which is always denoted by $X \overset{s}{\approx} Y$. If for any PPT algorithm $\mathcal{A}$ that $\left|\Pr[\mathcal{A}(1^\lambda, X) = 1] - \Pr[\mathcal{A}(1^\lambda, Y) = 1]\right|$ is negligible in λ, then we say that the two distributions are computationally indistinguishable, denoted by $X \overset{c}{\approx} Y$.

Matrix norms. For a vector $\boldsymbol{x}$, its Euclidean norm (also known as the ℓ_2 norm) is defined as $\|\boldsymbol{x}\| = (\sum_i x_i^2)^{1/2}$. For a matrix $\mathbf{R}$, we denote its ith column vector as $\boldsymbol{r}_i$, and use $\widetilde{\mathbf{R}}$ to denote its Gram-Schmidt orthogonalization. In addition,

- $\|\mathbf{R}\|$ denotes the Euclidean norm of $\mathbf{R}$, i.e., $\|\mathbf{R}\| = \max_i \|\boldsymbol{r}_i\|$.
- $s_1(\mathbf{R})$ denotes the spectral norm of $\mathbf{R}$, i.e., $s_1(\mathbf{R}) = \sup_{\|\boldsymbol{x}\|=1}\|\mathbf{R}\boldsymbol{x}\|$, with $\boldsymbol{x} \in \mathbb{Z}^m$.

We know the facts on the above norms: $\|\widetilde{\mathbf{R}}\| \le \|\mathbf{R}\| \le s_1(\mathbf{R}) \le \sqrt{k}\|\mathbf{R}\|$ and $s_1(\mathbf{R}|\mathbf{S}) \le \sqrt{s_1(\mathbf{R})^2 + s_1(\mathbf{S})^2}$, where k denote the number of columns of $\mathbf{R}$. Besides, we have the following lemma for the bounding spectral norm.

Lemma 2.1 ([24]) *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 $s_1(\mathbf{X}) \le c \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $\frac{2}{e^{\pi t^2}}$.*

At the same time, we rely on the following useful lemma on cover-free for our security proof.

Lemma 2.2 (Cover-Freeness [33]) *Let $\Delta_1, \cdots, \Delta_Q \subseteq [N]$ be randomly chosen subsets of size v. Let $v(\kappa) = \Theta(\kappa)$ and $N(\kappa) = \Theta(vQ^2)$. Then for all $i \in [Q]$, we have $\mathsf{Pr}\left[\Delta_i \backslash \left(\bigcup_{j\neq i}\Delta_j\right) \neq \emptyset\right] = 1 - 2^{-\Omega(\kappa)}$, where the probability is over the random choice of subsets $\Delta_1, \cdots, \Delta_Q$.*

2.2 Lattices Background

A lattice is a discrete additive subgroup of $\mathbb{R}^m$. Let $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m) \subset \mathbb{R}^m$ consists of m linearly independent vectors. The m-dimensional lattice Λ generated

by the basis $\mathbf{B}$ is $\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{B} \cdot \boldsymbol{c} = \sum_{i \in [m]} c_i \cdot \boldsymbol{b}_i : \boldsymbol{c} = (c_1, \ldots, c_m) \in \mathbb{Z}^m\}$. We will be interested in integer lattices, whose points have coordinateds in $\mathbb{Z}^m$. One of typical integer lattices is the q-ary lattice defined as follows: for any integer $q > 2$ and any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define $\Lambda_q(\mathbf{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \boldsymbol{x} = \mathbf{A}^\top \cdot \boldsymbol{e} \bmod q, \text{ for some } \boldsymbol{e} \in \mathbb{Z}_q^n\}$, $\Lambda_q^\perp(\mathbf{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \boldsymbol{x} = 0 \bmod q\}$, and the related shift $\Lambda_q^{\boldsymbol{u}}(\mathbf{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \boldsymbol{x} = \boldsymbol{u} \bmod q\}$.

The dual lattice of Λ, denoted as Λ^*, is defined to be $\Lambda^* = \{\boldsymbol{x} \in \mathbb{R}^m : \forall\ \boldsymbol{v} \in \Lambda,\ \langle \boldsymbol{x}, \boldsymbol{v} \rangle \in \mathbb{Z}\}$. By symmetry, $(\Lambda^*)^* = \Lambda$. For q-ary lattices $\Lambda_q(\mathbf{A})$ and $\Lambda_q^\perp(\mathbf{A})$, it holds $\Lambda_q^\perp(\mathbf{A}) = q \cdot (\Lambda_q(\mathbf{A}))^*$ and $\Lambda_q(\mathbf{A}) = q \cdot (\Lambda_q^\perp(\mathbf{A}))^*$.

The minimum distance $\lambda_1(\Lambda)$ of a lattice Λ is the length in the Euclidean ℓ_2 norm of the shortest nonzero vector: $\lambda_1(\Lambda) = \min_{0 \neq \mathbf{x} \in \Lambda} \|\boldsymbol{x}\|$. We write $\lambda_1^\infty(\Lambda)$ to denote the minimum length measured in the ℓ_∞ norm, which is defined as $\|\boldsymbol{x}\|_\infty = \max |x_i|$.

For an approximation factor $\gamma = \gamma(n) > 1$, we define the problem GapSVP_γ as follows: given a basis $\mathbf{B}$ of an m-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$ and a positive number d, distinguish between the case where $\lambda_1(\Lambda) \leq d$ and the case where $\lambda_1(\Lambda) \geq \gamma d$.

2.3 Gaussians on Lattices.

Let σ be any positive real number. The Gaussian distribution $\mathcal{D}_{\sigma,\boldsymbol{c}}$ with parameter σ and $\boldsymbol{c}$ is defined by probability distribution function $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = exp(-\pi \|\boldsymbol{x} - \boldsymbol{c}\|^2/\sigma^2)$. For any set $S \subseteq \mathbb{R}^m$, define $\rho_{\sigma,\boldsymbol{c}}(S) = \sum_{\boldsymbol{x} \in S} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$. The discrete Gaussian distribution $\mathcal{D}_{S,\sigma,\boldsymbol{c}}$ over S with parameter σ and $\boldsymbol{c}$ is defined by the probability distribution function $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})/\rho_{\sigma,\boldsymbol{c}}(S)$ for all $\boldsymbol{x} \in S$.

In [42], Micciancio and Regev introduced a useful quantity called smoothing parameter.

Definition 2.3 *For any m-dimensional lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$.*

Then, we have the following upper bound for the smoothing parameter.

Lemma 2.4 ([29]) *For any m-dimensional lattice Λ and real $\epsilon > 0$, we have $\eta_\epsilon(\Lambda) \leq \frac{\sqrt{\log(2m/(1+1/\epsilon))/\pi}}{\lambda_1^\infty(\Lambda^*)}$. Then for any $\omega(\sqrt{\log m})$ function, there is a negligible $\epsilon(m)$ for which $\eta_\epsilon(\Lambda) \leq \omega(\sqrt{\log m})/\lambda_1^\infty(\Lambda^*)$.*

Furthermore, we have the following useful facts from the literature.

Lemma 2.5 ([29] and Full Version of [40]) *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 2.6 *Let n, m, q are integers such that $m > 2n \log q$, and $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$ be arbitrary. 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 $\eta_\varepsilon(\Lambda_q^\perp(\mathbf{A}|\mathbf{AR})) \leq \omega(\sqrt{\log m})$.

Proof. By definition, we have $\Lambda_q(\mathbf{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \boldsymbol{x} = \mathbf{A}^\top \cdot \boldsymbol{e} \bmod q, \text{ for some } \boldsymbol{e} \in \mathbb{Z}_q^n\}$, and $\Lambda_q(\mathbf{A}|\mathbf{AR}) = \{\boldsymbol{x} \in \mathbb{Z}^{2m} : \boldsymbol{x} = (\mathbf{A}|\mathbf{AR})^\top \cdot \boldsymbol{e} \bmod q, \text{ for some } \boldsymbol{e} \in \mathbb{Z}_q^n\}$. Hence, any vector in $\Lambda_q(\mathbf{A}|\mathbf{AR})$ has a prefix in $\Lambda_q(\mathbf{A})$, i.e., for any $\boldsymbol{x} = (\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top)^\top \in \Lambda_q(\mathbf{A}|\mathbf{AR})$, it holds $\boldsymbol{x}_1 \in \Lambda_q(\mathbf{A})$. Besides, for any vector $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{Z}^{2m}$, we have $\|\boldsymbol{x}\|_\infty \geq \|\boldsymbol{x}_1\|_\infty$, according to the definition of infinite norm. Connecting this fact with Lemma 2.5, we conclude that for all but an at most q^{-n} fraction of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we have $\lambda_1^\infty(\Lambda_q(\mathbf{A}|\mathbf{AR})) > \lambda_1^\infty(\Lambda_q(\mathbf{A})) > q/4$.

The second part follows from the relation with λ_1 of the dual lattice as Lemma 2.4 and the fact that $\Lambda^\perp(\mathbf{A}|\mathbf{AR})^* = \Lambda(\mathbf{A}|\mathbf{AR})/q$, implying

$$\eta_\varepsilon(\Lambda_q^\perp(\mathbf{A}|\mathbf{AR})) \leq \omega(\sqrt{\log m})/\lambda_1^\infty(\Lambda^\perp(\mathbf{A}|\mathbf{AR})^*) \leq \omega(\sqrt{\log m}).$$

□

Lemma 2.7 ([29], Lemma 5.2) *Assume the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$, let $\epsilon \in (0, 1/2)$ and $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$. Then for $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, r}$, the distribution of $\boldsymbol{u} = \mathbf{A}^T\boldsymbol{e} \bmod q$ is within statistical distance 2ϵ of uniform over $\mathbb{Z}_q^n$.*

Furthermore, for any fixed $\boldsymbol{u} \in \mathbb{Z}_q^n$, let $\boldsymbol{t} \in \mathbb{Z}^m$ be an arbitrary solution to $\mathbf{A}\boldsymbol{t} = \boldsymbol{u} \bmod q$. Then the conditional distribution of $e \sim \mathcal{D}_{\mathbb{Z}^m, s}$ given $\mathbf{A}\boldsymbol{e} = \boldsymbol{u} \bmod q$ is exactly $\boldsymbol{t} + \mathcal{D}_{\Lambda^\perp, s, -\boldsymbol{t}}$.

Lemma 2.8 ([42], Lemma 4.4) *For any m-dimensional lattice Λ, $c \in \mathbf{R}^m$, real $\epsilon \in (0, 1)$ and $s \geq \eta_\epsilon(\Lambda)$, we have $\Pr_{\boldsymbol{x} \leftarrow \mathcal{D}_{\Lambda, s, \mathbf{c}}}[\|\boldsymbol{x} - \boldsymbol{c}\| > s\sqrt{m}] \leqslant \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-m}$.*

Lemma 2.9 (Smudging Lemma) *Let $n \in \mathbb{N}$. For any real $\sigma \geq \omega(\sqrt{\log n})$, and any $\boldsymbol{c} \in \mathbb{Z}^n$, it holds $\mathsf{SD}(\mathcal{D}_{\mathbb{Z}^n, \sigma}, \mathcal{D}_{\mathbb{Z}^n, \sigma, \boldsymbol{c}}) \leq \|\boldsymbol{c}\|/\sigma$.*

Lemma 2.10 (Noise Rerandomization [39]) *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 $\boldsymbol{b} \in \mathbb{Z}_q^m$ be arbitrary vector and $\boldsymbol{x}$ 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 $\mathsf{ReRand}(\mathbf{V}, \boldsymbol{b} + \boldsymbol{x}, r, \sigma)$ that outputs $\boldsymbol{b}' = \boldsymbol{b}\mathbf{V} + \boldsymbol{x}' \in \mathbb{Z}_q^\ell$ where the statistical distance of the discrete Gaussian $\mathcal{D}_{\mathbb{Z}^\ell, 2r\sigma}$ and the distribution of $\boldsymbol{x}'$ is within 8ϵ.*

Learning With Errors. The Learning with errors problem, or LWE, is the problem of determining a secret vector over $\mathbb{F}_q$ given a polynomial number of "noisy" inner products. The decision variant is to distinguish such samples from random. More formally, we define the problem as follows:

Definition 2.11 ([45]) *Let $n \geq 1$ and $q \geq 2$ be integers, and let χ be a probability distribution on $\mathbb{Z}_q$. For $\boldsymbol{s} \in \mathbb{Z}_q^n$, let $A_{\boldsymbol{s}, \chi}$ be the probability distribution on*

$\mathbb{Z}_q^n \times \mathbb{Z}_q$ *obtained by choosing a vector* $\boldsymbol{a} \in \mathbb{Z}_q^n$ *uniformly at random, choosing* $e \in \mathbb{Z}_q$ *according to* χ *and outputting* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e)$.

The decision $\mathsf{LWE}_{q,n,\chi}$ *problem is: for uniformly random* $\boldsymbol{s} \in \mathbb{Z}_q^N$, *given a poly(n) number of samples that are either (all) from* $A_{\boldsymbol{s},\chi}$ *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-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible if for all polynomial-time algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ solves the decision-$\mathsf{LWE}_{q,n,\chi}$ problem (over $\boldsymbol{s}$ and $\mathcal{A}$'s random coins) is negligibly close to $1/2$ as a function of n. The works of [18,44,45] show that the LWE assumption is as hard as (quantum or classical) solving GapSVP and SIVP under various parameter regimes.

2.4 Lattice Trapdoor and Gaussian Sampling

Gadget Matrix. We recall the "gadget matrix" $\mathbf{G}$ defined in [41]. The "gadget matrix" $\mathbf{G} = \mathbf{I}_n \otimes \boldsymbol{g}^\top \in \mathbb{Z}_q^{n \times n\lceil \log q \rceil}$ where $\boldsymbol{g}^\top = (1, 2, 4, ..., 2^{\lceil \log q \rceil - 1})$. We can also extend the column dimension to any $m \geq n\lceil \log q \rceil$ by padding $\mathbf{0}_{n \times m'}$ to the right for $m' = (m - n\lceil \log q \rceil)$, i.e., $\mathbf{G} = [\mathbf{I}_n \otimes \boldsymbol{g}^\top | \mathbf{0}_{n \times m'}] \in \mathbb{Z}_q^{n \times m}$.

Lemma 2.12 (Theorem 4.1, [41]) *Let* $q \geq 2$ *be any integer, and* $n, m \geq 2$ *be integers with* $m \geq n\lceil \log q \rceil$. *There is a full-rank (of columns) matrix* $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ *such that the lattice* $\Lambda_q^\perp(\mathbf{G})$ *has a publicly known trapdoor matrix* $\mathbf{T_G} \in \mathbb{Z}^{n \times m}$ *with* $\|\widetilde{\mathbf{T}}_\mathbf{G}\| \leq \sqrt{5}$, *where* $\widetilde{\mathbf{T}}_\mathbf{G}$ *is the Gram-Schmidt orthogonalization of* $\mathbf{T_G}$.

Theorem 2.13 (Trapdoor Generation [9, 41]) *There is a probabilistic polynomial-time algorithm* $\mathsf{TrapGen}(1^n, q, m)$ *that for all* $m \geq m_0 = m_0(n, q) = O(n \log q)$, *outputs* $(\mathbf{A}, \mathbf{T_A})$ *such that* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *is within statistical distance* 2^{-n} *from uniform, and* $\mathbf{T_A}$ *is a basis for* $\Lambda_q^\perp(\mathbf{A})$ *satisfying* $\|\mathbf{T_A}\| \leq O(n \log q)$ *and* $\|\widetilde{\mathbf{T}}_\mathbf{A}\| \leq O(\sqrt{n \log q})$, *where* $\widetilde{\mathbf{T}}_\mathbf{A}$ *denotes the Gram-Schmidt orthogonalization of* $\mathbf{T_A}$.

Lemma 2.14 (SampleLeft [3, 23]) *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* $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$ *and* $s \geq \|\widetilde{\mathbf{T}}_\mathbf{A}\| \cdot \omega(\sqrt{\log m})$. *Then there exists a* PPT *algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \mathbf{U}, s)$ *that outputs a matrix* $\mathbf{X} \in \mathbb{Z}_q^{2m \times \ell}$, *which is distributed statistically close to* $D_{\Lambda_q^\mathbf{U}(\mathbf{A}|\mathbf{B}),s}$.

Lemma 2.15 (SampleRight [41]) *Let* $q > 2$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *be a full rank matrix with* $m > n$, $\mathbf{R} \in \mathbb{Z}^{m \times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$, $y \in \mathbb{Z}_q$ *with* $y \neq 0$, *and* $s \geq \sqrt{5} \cdot s_1(\mathbf{R}) \cdot \omega(\sqrt{\log m})$. *Then there exists a* PPT *algorithm* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{R}, y, \mathbf{U}, s)$ *that outputs a matrix* $\mathbf{X} \in \mathbb{Z}_q^{2m \times \ell}$, *which is distributed statistically close to* $D_{\Lambda_q^\mathbf{U}(\mathbf{A}|\mathbf{A}\cdot\mathbf{R}+y\mathbf{G}),s}$, *where* $\mathbf{G}$ *is the gadget matrix.*

Lemma 2.16 (Leftover Hash Lemma [3]) *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* $\boldsymbol{e} \in \mathbb{Z}^m$*, the distribution* $(\mathbf{A}, \mathbf{AR}, \mathbf{R}^\top \boldsymbol{e})$ *is statistically close to the distribution* $(\mathbf{A}, \mathbf{B}, \mathbf{R}^\top \boldsymbol{e})$*.*

2.5 Partially Hiding Predicate Encryption

We recall the notation of partially hiding predicate encryption (PHPE) proposed by [35], which interpolates attribute-based encryption and predicate encryption. A Partially-Hiding Predicate Encryption scheme PHPE for a pair of private-public index spaces $\mathcal{X}, \mathcal{Y}$, a function class $\mathcal{F}$ mapping $\mathcal{X} \times \mathcal{Y}$ to $\{0,1\}$, and a message space $\mathcal{M}$, consists of four algorithms
$(\mathsf{PHPE.Setup}, \mathsf{PHPE.Enc}, \mathsf{PHPE.KeyGen}, \mathsf{PHPE.Dec})$:

- $\mathsf{PHPE.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathcal{M}) \to (\mathsf{PHPE.mpk}, \mathsf{PHPE.msk})$. The setup algorithm gets as input the security parameter λ and a description of $(\mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathcal{M})$ and outputs the public parameter PHPE.mpk, and the master key PHPE.msk.
- $\mathsf{PHPE.Enc}(\mathsf{PHPE.mpk}, (\boldsymbol{x}, \boldsymbol{y}), \mu) \to \mathsf{ct}_{\boldsymbol{y}}$. The encryption algorithm gets as input PHPE.mpk, a pair of private-public indexes $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ and a message $\mu \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct}_{\boldsymbol{y}}$.
- $\mathsf{PHPE.KeyGen}(\mathsf{PHPE.msk}, f) \to \mathsf{sk}_f$. The key generation algorithm gets as input PHPE.msk and a function $f \in \mathcal{F}$. It outputs a secret key sk_f.
- $\mathsf{PHPE.Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{\boldsymbol{y}}, \boldsymbol{y})) \to \mu \vee \perp$. The decryption algorithm gets as input the secret key sk_f, a function f, and a ciphertext $\mathsf{ct}_{\boldsymbol{y}}$ and the public part $\boldsymbol{y}$ of the attribute vector. It outputs a message $\mu \in \mathcal{M}$ or $\perp$.

Correctness. We require that for all $(\mathsf{PHPE.mpk}, \mathsf{PHPE.msk}) \leftarrow \mathsf{PHPE.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathcal{M})$, for all $(\boldsymbol{x}, \boldsymbol{y}, f) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{F}$ and for all $\mu \in \mathcal{M}$,

- For 1-queries, i.e., $f(\boldsymbol{x}, \boldsymbol{y}) = 1$, $\Pr[\mathsf{PHPE.Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{\boldsymbol{y}}, \boldsymbol{y})) \neq \mu] \leq \mathsf{negl}(\lambda)$.
- For 0-queries, i.e., $f(\boldsymbol{x}, \boldsymbol{y}) = 0$, $\Pr[\mathsf{PHPE.Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{\boldsymbol{y}}, \boldsymbol{y})) \neq \perp] \leq \mathsf{negl}(\lambda)$.

The formal definition of PHPE can be formally described as follows.

Definition 2.17 ((q_1, q_2)-Sel-SIM security) *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a* PHPE *for a pair of private-public index spaces* $\mathcal{X}, \mathcal{Y}$*, a function class* $\mathcal{F}$ *mapping* $\mathcal{X} \times \mathcal{Y}$ *to* $\{0,1\}$*, and a message space* $\mathcal{M}$*. For every stateful p.p.t. adversary* $\mathcal{A}$ *and a stateful p.p.t. simulator* Sim*, consider the following two experiments:*

$\mathbf{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(1^\lambda)$:	$\mathbf{Exp}^{\mathsf{ideal}}_{\Pi,\mathcal{A}}(1^\lambda)$:
1: $(\boldsymbol{x}, \boldsymbol{y}) \leftarrow \mathcal{A}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathcal{M})$	*1:* $(\boldsymbol{x}, \boldsymbol{y}) \leftarrow \mathcal{A}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathcal{M})$
2: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$	*2:* $\mathsf{mpk} \leftarrow \mathsf{Sim}_1(1^\lambda, \boldsymbol{y}, 1^{\lvert x\rvert})$
3: $(u, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$	*3:* $(u, L) \leftarrow \mathcal{A}^{\mathsf{Sim}_2(\cdot)}(\mathsf{mpk})$
4: Set $b = u$.	*4:* If there is ever a 1-key queried, set $(b, \mathsf{List}) = (u, L)$, else $(b, \mathsf{List}) = \perp$.
5: $\mathsf{ct}_{\boldsymbol{y}} \leftarrow \mathsf{Enc}(\mathsf{PH.mpk}, (\boldsymbol{x}, \boldsymbol{y}), b)$	*5:* $\mathsf{ct}_{\boldsymbol{y}} \leftarrow \mathsf{Sim}_3(b, \mathsf{st})$
6: $\alpha \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}_{\boldsymbol{y}})$	*6:* $\alpha \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sim}_4(\cdot)}}(\mathsf{ct}_{\boldsymbol{y}})$
7: Output $(\boldsymbol{x}, \boldsymbol{y}, u, \alpha)$	*7:* Output $(\boldsymbol{x}, \boldsymbol{y}, u, \alpha)$

Below we present some supplementary notes of the above description.

- *Here we use L to record all the predicate functions queried by the adversary.*
- *The oracle $\mathcal{O}$ handles the adversary's key query in the following way: if the query f is a 1-key, then the oracle sends (u, f) to Sim_4. Otherwise it sends $(\perp, f)$ to Sim_4. Upon receiving a simulated key from Sim_4, $\mathcal{O}$ forwards the key to the adversary.*
- *Both the adversary and the simulator are stateful, so they carry states during all stages of the interaction. For simplicity, we omit these states, except in the step 5 in the Ideal experiment. This is because Sim_3 is allowed to learn the predicate functions of the 1-keys queried in Step 3, capturing the information learned by the adversary in the real. We handle this by letting the experiment pass the list L to Sim_3 via the state.*

We say an adversary $\mathcal{A}$ is admissible if for all queries, there exists at most q_1 1-keys and q_2 0-keys with respect to the challenge index $(\boldsymbol{x}, \boldsymbol{y})$.[12] *The partially hiding predicate encryption scheme PHPE is said to be (q_1, q_2)-Sel-SIM secure if there exists a p.p.t. simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$ such that for every admissible p.p.t. adversary $\mathcal{A}$, the following two distributions are computationally indistinguishable:*

$$\left\{\mathbf{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \stackrel{c}{\approx} \left\{\mathbf{Exp}^{\mathsf{ideal}}_{\Pi,\mathsf{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

2.6 Key Homomorphic Evaluation

In this section, we first describe several preliminaries for the (key) homomorphic evaluation mechanism as introduced by [12]. Next, we describe a homomorphic evaluation procedure of functions of the form $\widehat{C} \circ \mathsf{IP}$ (defined in Section 5) by tweaking the schemes of [2, 35].

Three basic algorithms. The work [12] proposes three deterministic algorithms $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}, \mathsf{Eval}_{\mathsf{trap}}$ with the following properties:

- $\mathsf{Eval}_{\mathsf{pk}}(\{\mathbf{A}_i\}_{i\in[\ell]}, C)$: on input ℓ matrices $\mathbf{A}_1, \cdots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n\times m}$ and a function $C : \{0,1\}^\ell \to \{0,1\}$ with depth d, outputs a matrix $\mathbf{A}_C \in \mathbb{Z}_q^{n\times m}$;
- $\mathsf{Eval}_{\mathsf{ct}}(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\boldsymbol{u}_i\}_{i\in[\ell]}, \boldsymbol{y}, C)$ takes as input $\mathbf{A}_1, \cdots, \mathbf{A}_\ell$ and C as above, along with $\boldsymbol{y} \in \{0,1\}^\ell$ and ℓ vectors $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_\ell \in \mathbb{Z}_q^m$, and outputs a vector $\boldsymbol{u}_C \in \mathbb{Z}_q^m$;
- $\mathsf{Eval}_{\mathsf{trap}}(\{\mathbf{R}_i\}_{i\in[\ell]}, \mathbf{A}, C, \boldsymbol{y})$: on input matrices $\mathbf{R}_1, \cdots, \mathbf{R}_\ell$ over $\{-1,1\}^{m\times m}$, matrix $\mathbf{A}$ over $\mathbb{Z}_q^{n\times m}$, C as before, along with $\boldsymbol{y} \in \{0,1\}^\ell$, outputs a matrix $\mathbf{R}_C \in \{-1,1\}^{m\times m}$.

The algorithms satisfy the following properties:

[12] Compared with the definition in [2], our definition does not restrict the 1-query only before generating the challenge ciphertext.

– If $(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_\ell) \approx ((\mathbf{A}_1 + y_1 \cdot \mathbf{G})^\top \boldsymbol{s}, \cdots, (\mathbf{A}_\ell + y_\ell \cdot \mathbf{G})^\top \boldsymbol{s})$, then $\boldsymbol{u}_C \approx (\mathbf{A}_C + C(\boldsymbol{y}) \cdot \mathbf{G})^\top \boldsymbol{s}$.
– If $(\mathbf{A}_1, \cdots, \mathbf{A}_\ell) = (\mathbf{A}\mathbf{R}_1 - y_1 \cdot \mathbf{G}, \cdots, \mathbf{A}\mathbf{R}_\ell - y_\ell \cdot \mathbf{G})$ where $\mathbf{R}_1, \cdots, \mathbf{R}_\ell$ are matrices with small norms, then we have

$$\mathbf{A}_C = \mathbf{A}\mathbf{R}_C - C(\boldsymbol{y}) \cdot \mathbf{G}$$

where $\mathbf{R}_C$ is also a matrix with a small norm, roughly with a n^{2d} multiplicative blow-up.

The following lemma captures the above intuition.

Lemma 2.18 ([12]) *The algorithms* $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}, \mathsf{Eval}_{\mathsf{trap}}$ *satisfy the following properties. For all* $\mathbf{A}_1, \cdots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n \times m}$*, all* $\boldsymbol{y} \in \{0,1\}^\ell$*, all boolean circuit* C *of depth* d*, let* $\mathbf{A}_C = \mathsf{Eval}_{\mathsf{pk}}(\mathbf{A}_1, \cdots, \mathbf{A}_\ell, C)$*. Then,*

– *for all* $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_\ell \in \mathbb{Z}_q^m$ *and all* $\boldsymbol{s} \in \mathbb{Z}_q^n$*,*

$$\|\boldsymbol{u}_C - (\mathbf{A}_C + C(\boldsymbol{y}) \cdot \mathbf{G})^\top \boldsymbol{s}\|_\infty \leq O(\ell n \log q)^{O(d)} \cdot \max_{i \in [\ell]} \{\|\boldsymbol{u}_i - (\mathbf{A}_i + y_i \cdot \mathbf{G})^\top \boldsymbol{s}\|_\infty$$

where $\boldsymbol{u}_C = \mathsf{Eval}_{\mathsf{ct}}(\mathbf{A}_1, \cdots, \mathbf{A}_\ell, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_\ell, \boldsymbol{y}, C)$*.*
– *If* $(\mathbf{A}_1, \cdots, \mathbf{A}_\ell) = (\mathbf{A}\mathbf{R}_1 - y_1 \cdot \mathbf{G}, \cdots, \mathbf{A}\mathbf{R}_\ell - y_\ell \cdot \mathbf{G})$ *where* $\mathbf{R}_1, \cdots, \mathbf{R}_\ell \in \mathbb{Z}_q^{m \times m}$*, then we have*

$$\mathbf{A}_C = \mathbf{A}\mathbf{R}_C - C(\boldsymbol{y}) \cdot \mathbf{G},$$

where $\mathbf{R}_C = \mathsf{Eval}_{\mathsf{trap}}(\{\mathbf{R}_i\}_{i \in [\ell]}, \mathbf{A}, C, \boldsymbol{y})$ *and*

$$s_1(\mathbf{R}_C) \leq O(\ell n \log q)^{O(d)} \cdot \sqrt{m}.$$

Homomorphic evaluation of $\widehat{C} \circ \mathsf{IP}$. Recall that $\widehat{C} \circ \mathsf{IP} : \{0,1\}^t \times \{0,1\}^\ell \to \{0,1\}$ is the function defined as

$$(\widehat{C} \circ \mathsf{IP})(\boldsymbol{x}, \boldsymbol{y}) = \mathsf{IP}\left(\mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\right),$$

for $\boldsymbol{x}, \widehat{C}(\boldsymbol{y}) \in \mathbb{Z}_q^{t'}$, PT being the power-of-two function, and $t = t' \log q$.

Next, we describe a simple tweak of [35] that homomorphically evaluates our desired functions. Let $\widehat{C}_i$ denote the circuit computing the i-th output value of $\widehat{C}$. Then we define three deterministic algorithms $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}, \mathsf{Eval}_{\mathsf{trap}}$ for $\widehat{C} \circ \mathsf{IP}$ as:

– $\mathsf{Eval}_{\mathsf{pk}}(\{\mathbf{A}_i\}_{i \in [\ell]}, \{\mathbf{B}_i\}_{i \in [t]}, \widehat{C} \circ \mathsf{IP})$: on input $\ell + t$ matrices $\{\mathbf{A}_i\}_{i \in [\ell]}$, $\{\mathbf{B}_i\}_{i \in [t]}$, and circuit $\widehat{C} \circ \mathsf{IP}$, output a matrix $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} \in \mathbb{Z}_q^{n \times m}$ computed as follows:
  1. For $i \in [t']$, compute $\mathbf{A}_{\widehat{C}_i} = \mathsf{Eval}_{\mathsf{pk}}(\mathbf{A}_1, \cdots, \mathbf{A}_\ell, \widehat{C}_i)$;
  2. Rearrange the indices of matrices $\{\mathbf{B}_i\}_{i \in [t]}$ to $\{\mathbf{B}_{i',j'}\}_{i' \in [t'], j' \in [\log q]}$, where we express $i = (i'-1) \log q + j'$. We note that the mapping of $i \mapsto (i', j')$ can be any bijection[13], as long as it is consistent among all our evaluation algorithms. For $i' \in [t']$, compute $\mathbf{B}_{i'} = \sum_{j' \in [\log q]} \mathbf{B}_{i',j'} \cdot \mathbf{G}^{-1}(2^{j'-1} \cdot \mathbf{G})$

[13] For example, we can use the division with remainder, namely, $i' = \lfloor i / \log q \rfloor$ and $j' = i \bmod \log q$

3. Output $\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} = \sum_{i'\in[t']} \mathbf{A}_{\widehat{C}_{i'}} \cdot \mathbf{G}^{-1}(\mathbf{B}_{i'})$.

– $\mathsf{Eval}_{\mathsf{ct}}(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_i, \boldsymbol{v}_i\}_{i\in[t]}, \boldsymbol{y}, \widehat{C}\circ\mathsf{IP})$ on input matrices $\{\mathbf{A}_i\}_{i\in[\ell]}$, $\{\mathbf{B}_i\}_{i\in[t]}$ and $\widehat{C}\circ\mathsf{IP}$ as before, along with $\boldsymbol{y} \in \{0,1\}^\ell$ and $\ell+t$ vectors $\{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_i\}_{i\in[t]}$, output a vector $\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} \in \mathbb{Z}_q^m$ computed as follows:

1. For $i \in [t']$, compute $\boldsymbol{u}'_{\widehat{C}_i} = \mathsf{Eval}_{\mathsf{ct}}(\mathbf{A}_1, \cdots, \mathbf{A}_\ell, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_\ell, \boldsymbol{y}, \widehat{C}_i)$;
2. Rearrange the indices of the set $\{\mathbf{B}_i, \boldsymbol{v}_i\}_{i\in[t]}$ to $\{\mathbf{B}_{i',j'}, \boldsymbol{v}_{i',j'}\}_{i'\in[t'],j'\in[\log q]}$, where we express $i = (i'-1)\log q + j'$. For $i' \in [t']$, compute $\mathbf{B}'_{i'} = \sum_{j'\in[\log q]} \mathbf{B}_{i',j'} \cdot \mathbf{G}^{-1}(2^{j'-1}\cdot\mathbf{G})$;
3. For $i' \in [t']$, compute $\boldsymbol{v}_{i'} = \sum_{j'\in[\log q]} \mathbf{G}^{-1}(2^{j'-1}\cdot\mathbf{G})^\top \cdot \boldsymbol{v}_{i',j'}$;
4. Let $\boldsymbol{z} = \widehat{C}(\boldsymbol{y})$, and then output

$$\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} = \sum_{i'\in[t']} \left(z'_i \cdot \boldsymbol{v}_{i'} - \mathbf{G}^{-1}(\mathbf{B}_{i'})^\top \cdot \boldsymbol{u}'_{\widehat{C}_{i'}} \right).$$

– $\mathsf{Eval}_{\mathsf{trap}}(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}'_i\}_{i\in[t]}, \mathbf{A}, \widehat{C}\circ\mathsf{IP}, \boldsymbol{x}, \boldsymbol{y})$: on input matrices $\{\mathbf{R}_i\}_{i\in[\ell]}$, $\{\mathbf{R}_i\}_{i\in[t]}$, $\mathbf{A}$, circuit $\widehat{C}\circ\mathsf{IP}$, $\boldsymbol{x} \in \{0,1\}^t$, $\boldsymbol{y} \in \{0,1\}^\ell$, output a matrix $\mathbf{R}_{\widehat{C}\circ\mathsf{IP}}$ computed as follows:

1. For $i \in [t']$, compute $\mathbf{R}_{\widehat{C}_i} = \mathsf{Eval}_{\mathsf{trap}}(\mathbf{R}_1, \cdots, \mathbf{R}_\ell, \mathbf{A}, \boldsymbol{y}, \widehat{C}_i)$;
2. Set $\mathbf{B}_i = \mathbf{A}\cdot\mathbf{R}'_i - x_i\cdot\mathbf{G}$ for $i \in [t]$. Rearrange the indices of $\{\mathbf{B}_i\}_{i\in[t]}$ to $\{\mathbf{B}_{i',j'}\}_{i'\in[t'],j'\in[\log q]}$ as above. For $i' \in [t']$, compute $\mathbf{B}_{i'} = \sum_{j'\in[\log q]} \mathbf{B}_{i',j'}\cdot \mathbf{G}^{-1}(2^{j'-1}\cdot\mathbf{G})$;
3. Rearrange the indices of $\{\mathbf{R}'_i\}_{i\in[t]}$ to $\{\mathbf{R}'_{i',j'}\}_{i'\in[t'],j'\in[\log q]}$, where we express $i = (i'-1)\log q + j'$. For $i' \in [t']$, compute $\mathbf{R}'_{i'} = \sum_{j'\in[\log q]} \mathbf{R}'_{i',j'}\cdot \mathbf{G}^{-1}(2^{j'-1}\cdot\mathbf{G})$;
4. Let $\boldsymbol{z} = \widehat{C}(\boldsymbol{y})$, and then output

$$\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} = \sum_{i'\in[t']} \left(-\mathbf{R}_{\widehat{C}_{i'}} \mathbf{G}^{-1}(\mathbf{B}_{i'}) + z_{i'}\cdot\mathbf{R}'_{i'} \right).$$

Similar to the work [35], it is not hard to derive the following lemma.

Lemma 2.19 *The algorithms* $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}, \mathsf{Eval}_{\mathsf{trap}}$ *satisfy the following properties. For matrices in* $\mathbb{Z}_q^{n\times m}$ *namely* $\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_i\}_{i\in[t]}$*, any* $(\boldsymbol{x}, \boldsymbol{y}) \in \{0,1\}^t \times \{0,1\}^\ell$*, any boolean circuit* $\widehat{C}$ *of depth* d*, define* $\mathbf{A}_C = \mathsf{Eval}_{\mathsf{pk}}(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_i\}_{i\in[t]}, \widehat{C}\circ\mathsf{IP})$*. Then the following holds:*

– *For all vectors* $\{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_i\}_{i\in[t]}$ *and all* $\boldsymbol{s} \in \mathbb{Z}_q^n$*,*

$$\begin{aligned} &\|\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} - (\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\rangle \cdot \mathbf{G})^\top \boldsymbol{s}\|_\infty \\ \le\ & O(\ell n \log q)^{O(d)} \cdot \max_{i\in[\ell]} \left\{ \|\boldsymbol{u}_i - (\mathbf{A}_i + y_i\cdot\mathbf{G})^\top \boldsymbol{s}\|_\infty \right\}, \end{aligned}$$

where $\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_i, \boldsymbol{v}_i\}_{i\in[t]}, \boldsymbol{y}, \widehat{C}\circ\mathsf{IP})$.

– *If* $\mathbf{A}_i = \mathbf{A}\mathbf{R}_i - y_i \cdot \mathbf{G}$ *for* $i \in [\ell]$*, and* $\mathbf{B}_i = \mathbf{A}\mathbf{R}'_i - x_i \cdot \mathbf{G}$ *then we have*

$$\mathbf{A}_C = \mathbf{A}\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} + \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\rangle \cdot \mathbf{G}$$

where $\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{trap}}(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}'_i\}_{i\in[t]}, \mathbf{A}, \widehat{C}\circ\mathsf{IP}, \boldsymbol{x}, \boldsymbol{y})$ *and*

$$s_1(\mathbf{R}_{\widehat{C}\circ\mathsf{IP}}) \leq \max\{O(\log q)^2, O(\ell n \log q)^{O(d)}\} \cdot \sqrt{m}$$

2.7 Fully Homomorphic Encryption

A leveled homomorphic encryption scheme is a tuple of polynomial-time algorithms $(\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$:

- $\mathsf{FHE.KeyGen}(1^\lambda, 1^d, 1^k)$: This is a probabilistic algorithm that takes as input the security parameter, the depth bound for the circuit, the message length and outputs the secret key $\mathsf{FHE.sk}$.
- $\mathsf{FHE.Enc}(\mathsf{FHE.sk}, \mu)$: This is a probabilistic algorithm that takes as input the secret key and message and produces the ciphertext $\mathsf{FHE.ct}$.
- $\mathsf{FHE.Eval}(C, \mathsf{FHE.ct})$: This is a deterministic algorithm that takes as input a Boolean circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d and outputs another ciphertext $\mathsf{FHE.ct}_C$.
- $\mathsf{FHE.Dec}(\mathsf{FHE.sk}, \mathsf{FHE.ct})$: This is a deterministic algorithm that takes as input the secret key and a ciphertext and produces a bit or $\perp$.

Correctness. For $\mathsf{FHE.sk} \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d, 1^k)$ and C a circuit of depth d, we require that

$$\mathsf{Pr}\Big[\mathsf{FHE.Dec}\big(\mathsf{FHE.sk}, \mathsf{FHE.Eval}(C, \mathsf{FHE.Enc}(\mathsf{FHE.sk}, \mu))\big) = C(\mu)\Big] = 1 - \mathsf{negl}(\lambda).$$

Security. We require that for every stateful PPT adversary $\mathcal{A}$ and for all $d, k = \mathsf{poly}(\lambda)$, the following quantity

$$\mathsf{Pr}\left[b = b' \;:\; \begin{array}{l} \mathsf{FHE.sk} \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d, 1^k); \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(1^\lambda, 1^d, 1^k); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{FHE.ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE.sk}, \mu); \\ b' \leftarrow \mathcal{A}(\mathsf{FHE.ct}) \end{array}\right] - \frac{1}{2}$$

is negligible in λ.

Instantiating FHE from Learning with Errors. We rely on the following instantiation of FHE from LWE:

Theorem 2.20 ([8, 17, 19, 20, 30]) *There is a* FHE *scheme based on the* LWE *assumption such that, as long as* $q \geq O(\lambda^2)$:

1. $\mathsf{FHE.sk} \in \mathbb{Z}_q^t$ *for some* $t \in \mathsf{poly}(\lambda)$.

2. $\mathsf{FHE.ct} \in \{0,1\}^{\ell}$, *where* $\ell = \mathsf{poly}(\lambda, k, d, \log q)$.
3. $\mathsf{FHE.ct}_C \in \{0,1\}^{\ell}$.
4. *There exists an algorithm* $\mathsf{FHE.Scale}(p, q, \mathsf{FHE.ct})$*, which reduces the modulus of the input ciphertext* $\mathsf{FHE.ct}$ *from* p *to* q*. Moreover, let* B *be an upper bound of the noise in* $\mathsf{FHE.ct}$*. Then the noise of the resulting ciphertext can be upper bounded by* $B \cdot q/p + \mathsf{poly}(\lambda)$*, for some fixed polynomial* $\mathsf{poly}(\cdot)$*. Therefore, as long as* $B \cdot q/p$ *can be upper bounded by a polynomial (even if all* B, q, p *are super-polynomials), the noise in the resulting ciphertext can be upper bounded by a polynomial.*
5. *For any boolean circuit* C *of depth* d*,* $\mathsf{FHE.Eval}(C, \cdot)$ *is computed by a boolean circuit of depth* $\mathsf{poly}(d, \lambda, \log q)$.
6. $\mathsf{FHE.Dec}$ *on input* $\mathsf{FHE.sk}$ *and* $\mathsf{FHE.ct}_C$ *outputs a bit* $b \in \{0,1\}$*. If* $\mathsf{FHE.ct}_C$ *is an encryption of 1, then*

$$\sum_{i \in [t]} \mathsf{FHE.sk}[i] \cdot \mathsf{FHE.ct}_C[i] \in \big[\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B\big],$$

where B *is some upper bound of the noise in* $\mathsf{FHE.ct}_C$*. Otherwise, if* $\mathsf{FHE.ct}_C$ *is an encryption of 0, then*

$$\sum_{i \in [t]} \mathsf{FHE.sk}[i] \cdot \mathsf{FHE.ct}_C[i] \notin \big[\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B\big].$$

Notice that in the application of this work, B *is required to be polynomially bounded. We can achieve this by choosing appropriate* p *and* q *(and some other techniques).*
7. *Security relies on* $\mathsf{LWE}_{q,\theta(t),\chi}$.

3 Definitions of Functional Encryption

We first present the syntax of functional encryption.

Definition 3.1 (Functional Encryption) *Let* $\mathcal{F}$ *be a family of functions, where each* $f \in \mathcal{F}$ *is defined as* $f : \mathcal{U} \to \mathcal{Y}$*. A functional encryption (*FE*) scheme for* $\mathcal{F}$ *consists of four algorithms as follows.*

- $\mathsf{Setup}(1^{\lambda}, \mathcal{F})$: *Given as input the security parameter* λ *and a description of the function family* $\mathcal{F}$*, the algorithm outputs a pair of master public key and master secret key* $(\mathsf{mpk}, \mathsf{msk})$*. In the following algorithms,* mpk *is implicitly assumed to be part of their inputs.*
- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F})$*: Given as input the master secret key* msk *and a function* $f \in \mathcal{F}$*, the algorithm outputs a description key* sk_f.
- $\mathsf{Enc}(\mathsf{mpk}, u \in \mathcal{U})$: *Given as input the master public key and a message* $u \in \mathcal{U}$*, the algorithm outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: *Given as input the secret key* sk_f *and a ciphertext* ct*, the algorithm outputs a value* $y \in \mathcal{Y}$ *or* $\perp$ *if it fails.*

A functional encryption scheme is correct, if for all security parameter λ, any message $u \in \mathcal{U}$ and any function $f \in \mathcal{F}$, the decryption algorithm outputs the right outcome, i.e., $\Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_u) = f(u)] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is taken over $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda}, \mathcal{F})$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathsf{ct}_u \leftarrow \mathsf{Enc}(u)$.

More fine-grained syntax of FE. For FE with fine-grained syntax, each message u consists of two parts, namely $u := (\mathsf{x}, \mu)$, where $\mathsf{x} \in \mathcal{X}$ for some index (or attribute) space $\mathcal{X}$, and $\mu \in \mathcal{M}$ for message space $\mathcal{M}$. Additionally, each function f consists of two parts, namely, $f := (\mathsf{P}, g) \in \mathcal{P} \times \mathcal{G}$, where P is a predicate over the index space $\mathcal{X}$, and g is a function of the message space $\mathcal{M}$. The overall function acts as the following:

$$f(u) := \begin{cases} g(\mu) & \text{if } \mathsf{P}(\mathsf{x}) = 1 \\ \perp & \text{otherwise.} \end{cases} \tag{1}$$

Therefore, when decrypting the ciphertext $\mathsf{ct}_u = \mathsf{Enc}(\mathsf{mpk}, (\mathsf{x}, \mu))$ by $\mathsf{sk}_f = \mathsf{KeyGen}(\mathsf{msk}, (\mathsf{P}, g))$, the algorithm outputs $g(\mu)$ if $\mathsf{P}(\mathsf{x}) = 1$, and $\perp$ otherwise. Under this fine-grained syntax, we call a key $\mathsf{sk}_{f:=(\mathsf{P},g)}$ a 1-key with respect to an index x if $\mathsf{P}(\mathsf{x}) = 1$, or otherwise a 0-key. Intuitively, a 1-key is allowed to open the ciphertext, but a 0-key is not.

To differentiate the regular FE in Definition 3.1 and FE with the fine-grained syntax, we use different types of function classes, i.e., FE for $\mathcal{F}$ refers to the former and FE for $\mathcal{P} \times \mathcal{G}$ refers to the latter.

There are two important types of index studied in the literature – FE with private or public index, according to whether the index x is revealed to the decryption algorithm or not.

Our security notions simply follow from those in prior work [2, 14, 33]. It is important that for the simulation-based security, we can achieve a notion where any pre- and post-challenge key queries are allowed, while the prior work [2] requires the adversary to commit in one-shot to all the 1-key queries right after seeing the master public key.

Particularly, we present the formal definitions for private/public-index function encryption schemes. Besides, we discuss relations of function families between regular syntax FE and fine-grained one, and detailed comparisons of our definitions with other relevant work.

3.1 Functional Encryption with Private Index

FE with private index requires that the decryption algorithm cannot learn the index beyond whether it is satisfied, providing the strongest guarantee of privacy. We observe that in this setting, the formulation of FE with the fine-grained syntax is essentially equivalent to that of the regular FE as Definition 3.1. It is easy to see that the regular FE can realize the FE with the fine-grained syntax by using Equation (1) in Section 3. On the other hand, FE for class $\mathcal{P} \times \mathcal{G}$ can clearly realize FE for $\mathcal{G}$ in Definition 3.1 by including $\mathsf{x} := \varepsilon$, i.e., the empty string, and a predicate function P such that $\mathsf{P}(\varepsilon) = 1$.

Next, we notice that an important subclass FE for $\mathcal{P} \times \{I\}$ for the identity function I (even just for message space $\mathcal{M} = \{0, 1\}$) is also sufficient to realize FE for $\mathcal{P}$ as Definition 3.1. This is because for any $g \in \mathcal{P}$, we can use $\mathsf{sk}_{(g,I)}$ and $\mathsf{Enc}((\mu, 1))$ to simulate exactly the same effect of sk_g and $\mathsf{Enc}(\mu)$ of the regular FE. Therefore, it is already without loss of generality to adopt the form of FE

for $\mathcal{P} \times \{I\}$ when considering expressive classes $\mathcal{P}$, e.g., Agrawal [2] directly defined functional encryption in this form. This captures the strongly attribute-hiding predicate encryption (PE), a generalization of the prior notion of weakly attribute-hiding PE of [35], which only hides the index against 0-keys but not 1-keys.

Based on the above discussion, in the following, we just define the security notion of FE for $\mathcal{P} \times \{I\}$, as the goal of this work is to achieve general boolean circuits for $\mathcal{P}$. Then in the end of this section, we will discuss the relation between our definition and those in the prior work [2, 4, 14, 33].

Definition 3.2 ((q_1, q_2)-SA-SIM security) *Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an FE with* private index *for family $\mathcal{F} = \mathcal{P} \times \{I\}$ and index-message space $\mathcal{U} = \mathcal{X} \times \mathcal{M}$. Then for any stateful adversary $\mathcal{A}$ and stateful simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$, we consider the following two experiments:*

$\mathbf{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(1^\lambda)$:	$\mathbf{Exp}^{\mathsf{ideal}}_{\Pi,\mathsf{Sim}}(1^\lambda)$:
1: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$	*1:* $\mathsf{mpk} \leftarrow \mathsf{Sim}_1(1^\lambda)$
2: $\mathsf{x}^* \leftarrow \mathcal{A}(\mathsf{mpk})$	*2:* $\mathsf{x}^* \leftarrow \mathcal{A}(\mathsf{mpk})$
3: $(m, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$	*3:* $(m, L) \leftarrow \mathcal{A}^{\mathsf{Sim}_2(\cdot)}(\mathsf{mpk})$
4: Set $b = m$	*4:* If there is ever a 1-key queried, set $(b, \mathsf{List}) = (m, L)$, else $(b, \mathsf{List}) = \bot$.
5: $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{x}^*, b)$	*5:* $\mathsf{ct} \leftarrow \mathsf{Sim}_3(b, \mathsf{List})$
6: $\alpha \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct})$	*6:* $\alpha \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sim}_4(\cdot)}}(\mathsf{ct})$
7: Output $(\mathsf{x}^*, m, \alpha)$	*7:* Output $(\mathsf{x}^*, m, \alpha)$

Below we present some supplementary notes of Definition 3.2.

- *Here we use L to record all the functions, i.e., $(\mathsf{P}, I) \in \mathcal{P} \times \{I\}$ of the 1-keys (with respect to x^*) queried by the adversary in Step 3. We assume that the simulator carry states, so for any $(\mathsf{P}, I) \in L$, Sim_3 knows the secret key $\mathsf{sk}_{(\mathsf{P},I)}$ generated by Sim_2.*
- *The oracle $\mathcal{O}$ handles the adversary's key query in the following way: if the query (P, I) is a 1-key with respect to x^*, then the oracle sends $(m, (\mathsf{P}, I))$ to Sim_4, or else it sends $(\bot, (\mathsf{P}, I))$. Then Sim_4 sends a simulated key to $\mathcal{O}$, and $\mathcal{O}$ forwards the key to the adversary.*
- *Both the adversary and the simulator are stateful, so they carry states during all stages of the interaction. For simplicity, we omit these states, except in the step 5 in the Ideal experiment. This is because the simulator (i.e., Sim_3) is allowed to learn the predicate functions of the 1-keys queried in Step 3, capturing the information learned by the adversary in the real. We handle this by letting the experiment pass the list L to Sim_3 via the state.*

We say an adversary $\mathcal{A}$ is admissible if for all queries, there exist at most q_1 1-keys and q_2 0-keys with respect to the challenge index x^*. The FE scheme

Π is said to be (q_1, q_2)-$\mathsf{SA\text{-}SIM}$ secure if there exists a PPT *simulator* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$ *such that for any admissible* PPT *adversary $\mathcal{A}$, the following two distributions are computationally indistinguishable:*

$$\left\{\mathbf{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathbf{Exp}^{\mathsf{ideal}}_{\Pi,\mathsf{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

Remark 3.3 *We call the adversary's queries made in Step 3 as* pre-challenge *key queries, and those made in Step 6 as* post-challenge *key queries. In the literature, there have been different names such as non-adaptive/adaptive key queries. To distinguish adaptiveness from key queries and index, in this work we always use non-adaptive/adaptive regarding the challenge index, and pre/post-challenge regarding the key queries.*

Comparison with Relevant Work. We compare our definition with other notions in the most relevant work [2, 4, 14, 33].

- This work and the prior work [2, 33] focus on the simulation-based security with respect to **one challenge** ciphertext for general functions. As pointed out by [14], simulation-based security for general functions is impossible for multiple challenge ciphertexts.
- Our definition and that of [2] work on the fine-grained syntax for $\mathcal{C} \times \{I\}$, yet our notion of semi-adaptive security is strictly stronger in the following sense: we only require that the adversary commits to the challenge index right after the mpk. After this step, he is allowed to make both pre- and post-challenge key queries for both 1-keys and 0-keys at any order he wishes. On the contrary, the work [2] requires that the adversary commits to both challenge index and all the challenge 1-key queries right after the mpk. After this step, no more 1-key queries are allowed.
- Both our work and [2] can achieve (Q, poly)-security for any polynomially bounded Q, and an unbounded polynomial poly. This is strictly stronger than the prior predicate encryption of [35], which only achieves $(0, \mathsf{poly})$-security and the notion of the weakly attribute hiding, i.e., the index is hidden only for 0-keys.
- The work [33] can achieve fully-adaptive simulation-based security, but the number of key queries is more restricted, i.e., (Q, Q)-security for any polynomially bounded Q.
- The work [4] showed that simulation-based security notion is impossible for general functions if the adversary can query an unbounded number of pre-challenge 1-keys. Therefore, it is not possible to extend the feasibility results of our work and [2, 33] to $(\mathsf{poly}, \mathsf{poly})$-security. We note that the lower bound of [4] does not apply to the 0-keys, and therefore there is no contradiction with the feasibility results of (Q, poly)-security achieved in [2] and this work.

3.2 Functional Encryption with Public Index

FE with public index does not prohibit the decryption algorithm from learning the index from the ciphertext, regardless of whether it is satisfied. In fact, this

notion has been studied extensively in the literature, capturing many useful schemes for different expressions of $\mathcal{P} \times \mathcal{G}$. For example, IBE can be captured by setting $\mathcal{G} = \{I\}$ and $\mathcal{P}$ as the class of identify comparison functions; ABE can be captured by setting $\mathcal{G} = \{I\}$ and $\mathcal{P}$ as the class of general predicate functions; in a recent work [1], Abdalla et al. [1] considered $\mathcal{P}$ for the identity comparison functions or general predicates, and $\mathcal{G} = \mathsf{IP}$ the inner product functions. These naturally define IB-FEIP and AB-FEIP, respectively.

Even though it is possible to construct FE with public index via the feasibility result of FE with private index (e.g., [2]), in this work we aim at much more efficient constructions for the same classes as Abdalla et al. [1], i.e., IB-FEIP and AB-FEIP. We achieve the indistinguishability based security as we present next.

Definition 3.4 ((q_1, q_2)-IND security) *Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a* public index FE *scheme for a function class $\mathcal{F} = \mathcal{P} \times \mathcal{G}$ and a encrypted space $\mathcal{U} = \mathcal{X} \times \mathcal{M}$. For any security parameter λ, and any stateful adversary $\mathcal{A}$, we define the following security experiment for $\beta \in \{0, 1\}$ as follows.*

Experiment $\mathsf{IND}^{\Pi}_{\beta}(1^{\lambda}, \mathcal{A})$*:*
(Sel)*:* $\mathsf{x}^* \leftarrow \mathcal{A}(1^{\lambda}, \mathcal{P}, \mathcal{G}, \mathcal{X}, \mathcal{M})$
1: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda}, \mathcal{P}, \mathcal{G}, \mathcal{X}, \mathcal{M})$
(SA)*:* $\mathsf{x}^* \leftarrow \mathcal{A}(\mathsf{mpk})$
2: $(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk})$
(AD)*:* $\mathsf{x}^* \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk})$
3: $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{x}^*, \mu_{\beta})$
4: $\alpha \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}^*)$
5: Output $\beta' \in \{0, 1\}$

Below we present some supplementary notes of Definition 3.4.

- *The adversary is stateful, so it carries states during all stages of the interaction. For simplicity, we omit these states.*
- *The adversary is admissible if the following holds:*
  - *for the challenge messages (μ_0, μ_1) and index x^*, all key queries for $f = (\mathsf{P}, g)$ satisfy the condition $g(\mu_0) = g(\mu_1)$ when $\mathsf{P}(\mathsf{x}^*) = 1$;*
  - *the adversary makes at most q_1 1-keys and q_2 0-keys.*
- *There will be only one step among* {Sel, SA, AD} *being selected by the adversary, capturing the selective, semi-adaptive, or adaptive indistinguishable experiment, respectively.*

We define the advantage of $\mathcal{A}$ in the above experiment to be

$$\mathbf{Adv}^{\mathsf{IND}}_{\Pi, \mathcal{A}}(\lambda) = \left| \Pr[\mathsf{IND}^{\Pi}_0(1^{\lambda}, \mathcal{A}) = 1] - \Pr[\mathsf{IND}^{\Pi}_1(1^{\lambda}, \mathcal{A}) = 1] \right|.$$

An FE *scheme is said to be selevtive/semi-adaptive/adaptive (q_1, q_2)-*IND*-secure if for any* PPT *admissible adversary $\mathcal{A}$, we have $\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{FE}, \mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. Moreover, whether the scheme is selective, semi-adaptive, or adaptive security depends on the step where x^* is chosen by the adversary.*

The case of $\mathsf{IB} \times \mathcal{G}$. We refer this special case as IB-FE as the predicate $\mathcal{P}$ contains only the comparison function, similar to the case of IBE. In this case, $(1, \mathsf{poly})$-AD-IND security is the best we can achieve, because there is only one possible 1-key for each challenge index.

Comparison with Relevant Work. The most relevant work to this work in the public index is a very recent work of [1]. For lattice-based constructions, the work [1] achieves selective-IND security for the class $\mathsf{IB} \times \mathsf{IP}$ in the plain model, and adaptive-IND security for the same class in the random oracle model. For more general predicate classes (e.g., general circuits), they cannot even achieve the selective-IND security under known techniques prior to this work. Besides, there is a slight difference between our definition of IND-security and that of [1]. In particular, our definition is much more fine-grained, i.e., we separate the challenge index from the challenge message, rather than putting them together as [1].

For this work, we are able to achieve the adaptive-IND for the class $\mathsf{IB} \times \mathsf{IP}$ and selective-IND for the class $\mathcal{C} \times \mathsf{IP}$, where $\mathcal{C}$ is the general boolean circuits. Both our constructions are in the plain model.

4 Our New Two-Stage Sampling Method

In this section, we present our key technical contribution – a new two-stage sampling method. At a high level, we would like to sample the following two-stage distribution: (1) in the first stage, a random matrix $\mathbf{A}$ and a random vector $\boldsymbol{u}$ are sampled, and (2) in the second stage, an arbitrary small-norm matrix $\mathbf{R}$ is given, and then some short vector $\boldsymbol{y}$ is sampled conditioned on $[\mathbf{A}|\mathbf{AR}]\boldsymbol{y} = \boldsymbol{u}$. The distribution then outputs $(\mathbf{A}, \mathbf{AR}, \boldsymbol{u}, \boldsymbol{y})$.

For a simpler case where there is no $\mathbf{R}$, this task is achievable via the following GPV two-stage sampling technique:

Lemma 4.1 ([29]) *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:*

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

Intuitively, we can pre-sample a short vector $\boldsymbol{y}$ from an appropriate Gaussian distribution and then set $\boldsymbol{u} = \mathbf{A}\boldsymbol{y}$. By the indistinguishability as Lemma 4.1, we can sample the desired distribution with or without the trapdoor of $\mathbf{A}$ as desired.[14] Moreover, this idea can be generalized to achieve a weaker version of our task where $\mathbf{R}$ is given in the first stage. The generalized idea has been

[14] To sample $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}), s}$, the current sampling algorithm requires that $s > \|\widetilde{\mathbf{T}}_{\mathbf{A}}\| \omega(\sqrt{\log m})$. According to the best known (to our knowledge) trapdoor generation, the smallest s we can sample would be $\omega(\sqrt{n \log q} \cdot \sqrt{\log m})$, which is much larger than the required bound for Lemma 4.1.

explored in the context of functional encryption (more precisely PHPE) by prior work [2], yet the technique however, would inherently require to know $\mathbf{R}$ in the first stage, resulting in a weak notion of very selective PHPE, where the adversary needs to commit to the challenge index and 1-key query at the beginning.

To break this limitation, we design a new two-stage sampling method that uses smudging noise over keys. Below we first present the two-stage sampling method and then explain the idea behind it.

For any integers $m > n \geq 1, q \geq 2$, we consider the following two procedures:

Sampler-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 $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$;
   - Sample a random vector $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n$;
2. Stage 2:
   - Sample a random $\boldsymbol{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \rho}$;
   - Compute $\boldsymbol{z} = \boldsymbol{u} - \mathbf{A}\boldsymbol{x} \pmod q$;
   - Sample a vector $\boldsymbol{z}' = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{z}}(\mathbf{A}|\mathbf{AR}), s}$, satisfying $(\mathbf{A}|\mathbf{AR}) \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} = \boldsymbol{z} \pmod q$;
   - Set $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} + \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} \in \mathbb{Z}^{2m}$, satisfying $(\mathbf{A}|\mathbf{AR})\boldsymbol{y} = \boldsymbol{u} \pmod q$;
   - Output the tuple $(\mathbf{A}, \mathbf{AR}, \boldsymbol{y}, \boldsymbol{u})$.

The Sampler-1$(\mathbf{R}, \rho, s)$ can be implemented efficiently given the trapdoor $\mathbf{T}_{\mathbf{A}}$ of $\mathbf{A}$, using the SampleLeft algorithm as Lemma 2.14 (with larger parameters of s than the required bound in Lemma 4.1). Next we present another way to sample the distribution without the need of the trapdoor.

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 $\boldsymbol{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sqrt{\rho^2 + s^2}}$, and set $\boldsymbol{u} = \mathbf{A}\boldsymbol{x} \pmod q$;
2. Stage 2:
   - Sample a random vector $\boldsymbol{z}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s}$;
   - Compute a vector $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} - \mathbf{R}\boldsymbol{z}_2 \\ \boldsymbol{z}_2 \end{pmatrix}$, satisfying $(\mathbf{A}|\mathbf{AR})\boldsymbol{y} = \boldsymbol{u} \pmod q$;
   - Output the tuple $(\mathbf{A}, \mathbf{AR}, \boldsymbol{y}, \boldsymbol{u})$.

In a nutshell, this algorithm first pre-samples a (larger) $\boldsymbol{x}$ and sets $\boldsymbol{u} = \mathbf{A}\boldsymbol{x}$, without knowing $\mathbf{R}$. In the second stage when $\mathbf{R}$ is given, it samples a smaller $\boldsymbol{z}_2$ and adjusts $\boldsymbol{y}$ accordingly. Intuitively, the larger $\boldsymbol{x}$ servers as the smudging noise that "overwrites" the effect of $\mathbf{R}\boldsymbol{z}_2$ as long as the norm of $\boldsymbol{x}$ is super-polynomially larger. This would hide the information of $\mathbf{R}$, which needs to be kept secret as required by the proof framework in prior work [2,3]. We formalize this intuition by the following theorem.

Theorem 4.2 *For integers $q \geq 2$, $n \geq 1$, sufficiently large $m = O(n \log q)$, any $\mathbf{R} \in \mathbb{Z}^{m \times m}$, $s > \omega(\sqrt{\log m})$, and $\rho \geq s\sqrt{m}\|\mathbf{R}\| \cdot \lambda^{\omega(1)}$, the output distributions $(\mathbf{A}, \mathbf{AR}, \boldsymbol{y}, \boldsymbol{u})$ of the above two procedures are statistically close.*

Proof. Our high-level proof idea is to introduce an additional two-stage sampling algorithm Sampler-3, and then prove it statistically indistinguishable from both Sampler-1 and Sampler-2. Below, we describe the algorithm Sampler-3$(\mathbf{R}, \rho, s)$.

Sampler-3$(\mathbf{R}, \rho, s)$: Given a matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$ and two values $\rho, s \in \mathbb{R}$ as input, this sampler conducts the following steps in two stages.

1. Stage 1: Sample a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$;
2. Stage 2:
   - Sample two random vectors $\boldsymbol{x}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sqrt{\rho^2 + s^2}}$, $\boldsymbol{z}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s}$;
   - Compute $\boldsymbol{u} = (\mathbf{A}|\mathbf{AR}) \begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{z}_2 \end{pmatrix} (\mathrm{mod} q)$, and denote $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{z}_2 \end{pmatrix} \in \mathbb{Z}^{2m}$;
   - Output a tuple $(\mathbf{A}, \mathbf{AR}, \boldsymbol{y}, \boldsymbol{u})$.

Claim 4.3 *For the parameters in the statement of Theorem 4.2, the output distributions of* Sampler-1 *and* Sampler-3 *are statistically close.*

Proof. We first observe that in Sampler-3, the $\boldsymbol{x}'$ component can be decomposed into $\boldsymbol{x} + \boldsymbol{z}_1$ (within a negligible statistical distance), where $\boldsymbol{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \rho}$ and $\boldsymbol{z}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s}$. The decomposition holds as we have $\rho > s > \eta_\varepsilon(\mathbb{Z}^m)$ for some $\varepsilon = \mathsf{negl}(\lambda)$.

Next, we prove a generalization of Lemma 4.1 that the following two distributions are statistically close:

- D_1: $\left(\mathbf{A}, \mathbf{AR}, \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}, \boldsymbol{u}'\right)$: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{u}' \xleftarrow{\$} \mathbb{Z}_q^n$, $\begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{u}'}(\mathbf{A}|\mathbf{AR}), s}$.
- D_2: $\left(\mathbf{A}, \mathbf{AR}, \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}, \boldsymbol{u}'\right)$: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m}, s}$,
  $\boldsymbol{u}' = (\mathbf{A}|\mathbf{AR}) \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} \bmod q$.

This simply follows from Lemmas 2.6 and 2.7 – for all but q^{-n} fraction of $\mathbf{A}$, we have $\eta_\epsilon(\Lambda^\perp(\mathbf{A}|\mathbf{AR})) \leq \omega(\sqrt{\log m}) < s$; for such an $\mathbf{A}$, the distribution of $(\mathbf{A}|\mathbf{AR}) \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}$ is uniformly random over $\mathbb{Z}_q^n$, and the conditional distribution

of $\begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}$ given the constraint is $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}|\mathbf{AR}),s}$. Thus, we conclude that D_1 and D_2 are statistically close.

The above indistinguishability implies directly that the following two distributions are as well statistically indistinguishable:

- D_1': $\left(\mathbf{A}, \mathbf{AR}, \begin{pmatrix} \boldsymbol{z}_1 + \boldsymbol{x}' \\ \boldsymbol{z}_2 \end{pmatrix}, \boldsymbol{u}' + \mathbf{A}\boldsymbol{x}'\right)$: $\boldsymbol{x}' \leftarrow D_{\mathbb{Z}^m,\rho}$; the other random variables are sampled the same way as D_1.
- D_2': $\left(\mathbf{A}, \mathbf{AR}, \begin{pmatrix} \boldsymbol{z}_1 + \boldsymbol{x}' \\ \boldsymbol{z}_2 \end{pmatrix}, \boldsymbol{u}' + \mathbf{A}\boldsymbol{x}'\right)$: $\boldsymbol{x}' \leftarrow D_{\mathbb{Z}^m,\rho}$; the other random variables are sampled the same way as D_2.

As one can apply the same randomized procedure F such that $D_1' = F(D_1)$ and $D_2' = F(D_2)$, we conclude that $\mathsf{SD}(D_1', D_2') \leq \mathsf{SD}(D_1, D_2) < \mathsf{negl}(\lambda)$.

Finally, by change of variable with $\boldsymbol{u} = \boldsymbol{u}' + \mathbf{A}\boldsymbol{x}'$, we can easily see that the marginal distribution of $\boldsymbol{u}$ is still uniformly random in D_1', i.e., ($\boldsymbol{u}'$ serves as a one-time pad). Then it is not hard to see that D_1' is distributed identical as Sampler-1 and D_2' is distributed statistically close to Sampler-3. This concludes the proof of the claim. □

Claim 4.4 *For the parameters in the statement of Theorem 4.2, the output distributions of* Sampler-2 *and* Sampler-3 *are statistically close.*

Proof. We first observe that for both Sampler-2 and Sampler-3, the component $\boldsymbol{u}$ can be determined (deterministically) from the first three components $(\mathbf{A}, \mathbf{AR}, \boldsymbol{y})$. Therefore, it suffices for us just to prove statistical closeness for the first three components.

We next note that $\mathbf{A}$ is uniformly random and independent with the component $\boldsymbol{y}$ in both Sampler-2 and Sampler-3. Therefore, it remains to show that the distributions of $\boldsymbol{y}$ in these two algorithms are statistically close.

In Sampler-2, we have $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} - \mathbf{R}\boldsymbol{z}_2 \\ \boldsymbol{z}_2 \end{pmatrix}$, and in Sampler-3 we have $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z}_2 \end{pmatrix}$. As $\rho \geq s\sqrt{m}\|\mathbf{R}\| \cdot \lambda^{\omega(1)}$, by the smudging lemma (i.e., Lemma 2.9) and the Gaussian tail bound (i.e., Lemma 2.8), these two distributions are statistically close. This concludes the proof of the claim. □

The proof of this theorem follows directly from the above two claims. □

5 Constructions of PHPE and FE with Private Input

In this section, we present three constructions of partially hiding predicate encryption scheme PHPE. Particularly, we first construct a basic $(1, \mathsf{poly})$-Sel-SIM secure PHPE in Section 5.1. Then, we upgrade our basic scheme to a (Q, poly)-Sel-SIM secure PHPE for any polynomially bounded Q and general key queries in Section 5.2. In Section 5.3, we show how to obtain a (Q, poly)-SA-SIM secure PHPE via a simple transformation. Finally, we present the construction of (Q, poly)-SIM-secure Functional Encryption with private input in Section 6.

Throughout the whole section, we will work on the function class $\mathcal{F}$ as described below. Before presenting the class, we first define three basic functions.

Definition 5.1 *Let $t \in \mathbb{N}, q \in \mathbb{N}, t' \in \mathbb{N}$ such that $t = t' \log q$. Define the function $\mathsf{PT} : \{0,1\}^t \to \mathbb{Z}_q^{t'}$ as: on input $\boldsymbol{x} \in \{0,1\}^t$, first parse the vector $\boldsymbol{x}$ into a bit matrix $\{x'_{i,j}\}_{i\in[t'],j\in[\log q]}$. The function then computes $\boldsymbol{z} = (z_1, \ldots, z_{t'})^\top$ as $z_i = \sum_{j\in[\log q]} x'_{i,j} \cdot 2^{j-1}$ for $i \in [t']$ and outputs $\boldsymbol{z} \in \mathbb{Z}_q^{t'}$.*

Definition 5.2 *Let $t' \in \mathbb{N}$ be the dimension of vectors, q be some modulus, and $\gamma \in \mathbb{Z}_q$ be some parameter. Define $\mathsf{IP} : \mathbb{Z}_q^{t'} \times \mathbb{Z}_q^{t'} \to \mathbb{Z}_q$ be the inner product modulo q, and $\mathsf{IP}_\gamma : \mathbb{Z}_q^{t'} \times \mathbb{Z}_q^{t'} \to \{0,1\}$ be function such that $\mathsf{IP}_\gamma(\boldsymbol{x}, \boldsymbol{y}) = 1$ if and only if $\gamma = \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y})$ for inputs $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^{t'}$.*

Intuitively, PT acts as the "power-of-two" function that maps $\{0,1\}^t$ to $\mathbb{Z}_q^{t'}$, and IP_γ acts as the comparison function between the parameter γ and the inner product of the inputs.

Function Class $\boldsymbol{\mathcal{F}}$. We consider functions of the following form. Any function in the class $\mathcal{F}$, namely $C : \{0,1\}^t \times \{0,1\}^\ell \to \{0,1\}$ can be described as $\widehat{C} \circ \mathsf{IP}_\gamma$, where $\widehat{C} : \{0,1\}^\ell \to \{0,1\}^{t'}$ is a boolean circuit of depth d, $t' \log q = t$, and $\gamma \in \mathbb{Z}_q$. More formally, for $\boldsymbol{x} \in \{0,1\}^t$ and $\boldsymbol{y} \in \{0,1\}^t$, the function is defined as

$$(\mathsf{IP}_\gamma \circ \widehat{C})(\boldsymbol{x}, \boldsymbol{y}) = \mathsf{IP}_\gamma\left(\mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\right).$$

Similarly, we define a relevant function $(\mathsf{IP} \circ \widehat{C}) : \{0,1\}^t \times \{0,1\}^\ell \to \mathbb{Z}_q$ as

$$(\mathsf{IP} \circ \widehat{C})(\boldsymbol{x}, \boldsymbol{y}) = \mathsf{IP}\left(\mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\right) = \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\rangle (\mathrm{mod} q).$$

Notice that our formulation is slightly different from that of the prior work [2, 35], which directly defined the input $\boldsymbol{x}$ in the domain $\mathbb{Z}_q^{t'}$. In Section 2.6, we show that this formulation can also achieve the same effect as the prior work [2, 35] with a simple tweak. Thus, it is without loss of generality to define functions in this way. In fact, our modified formulation is for the need of the transformation (from selective-security to semi-adaptive security) in Section 5.3, which requires to work on a small input base, e.g., $\{0,1\}$. We notice that both our selective PHPE and the scheme of [2] require a super-polynomial q, so without the modification of the input space, the selective scheme would not be compatible with the transformation.

5.1 $(1, \mathsf{poly})$-Partially Hiding Predicate Encryption

Our basic construction of PHPE is essentially the same as that of Agrawal [2] (her basic construction), except that we adopt our new sampling algorithm in Section 4 for the key generation. Our scheme achieves $(1, \mathsf{poly})$-Sel-Sim security as the Definition 2.17, where one 1-key pre-challenge query is allowed. This is

stronger than the $(1, \mathsf{poly})$-very-selective scheme of Agrawal [2], which requires the adversary to commit to both his challenge index and function of the 1-key query at the beginning of the experiment. Below we present the construction.

$\mathsf{PH.Setup}(1^\lambda, 1^t, 1^\ell, 1^d)$: Given as input the security parameter λ, the length of the private and public indices, t and ℓ respectively, and the depth of the circuit family d, the algorithm does the following steps:

1. Choose public parameters (q, ρ, s) as described in the following parameter setting paragraph.
2. Choose random matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ for $i \in [\ell], \mathbf{B}_j \in \mathbb{Z}_q^{n\times m}$ for $j \in [t]$, and $\mathbf{P} \in \mathbb{Z}_q^{n\times m}$.
3. Sample $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$.
4. Output the public and master secret keys.

$$\mathsf{PH.mpk} = (\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \mathbf{A}, \mathbf{P}), \mathsf{PH.msk} = (\mathbf{T}_\mathbf{A}).$$

$\mathsf{PH.KeyGen}(\mathsf{PH.msk}, \widehat{C} \circ \mathsf{IP}_\gamma)$: Given as input a circuit description $\widehat{C} \circ \mathsf{IP}_\gamma$ and the master secret key, the algorithm does the following steps:

1. Let $\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_\mathsf{pk}(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \widehat{C} \circ \mathsf{IP})$.
2. Sample matrix $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, and let $\mathbf{U} = \mathbf{P} - \mathbf{A}\mathbf{J} (\mathrm{mod} q)$.
3. Sample $\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\mathbf{G}, \mathbf{T}_\mathbf{A}, \mathbf{U}, s)$ for parameter s, i.e., the equation holds for $[\mathbf{A}|\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\mathbf{G}] \cdot \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{U} (\mathrm{mod} q)$.
4. Let $\mathbf{K} = \begin{bmatrix}\mathbf{J} + \mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix}$, and output $\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma} = \mathbf{K}$.

$\mathsf{PH.Enc}(\mathsf{PH.mpk}, (\boldsymbol{x}, \boldsymbol{y}), \mu)$: Given as input the master public key, the private attributes $\boldsymbol{x} \in \{0,1\}^t$, public attributes $\boldsymbol{y} \in \{0,1\}^\ell$ and message $\mu \in \{0,1\}$, the algorithm does the following steps:

1. Sample $\boldsymbol{s} \leftarrow D_{\mathbb{Z}^n,s_B}$ and error terms $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_B}$ and $\boldsymbol{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}$.
2. Let $\boldsymbol{b} = [0, \cdots, 0, \lceil q/2 \rceil \mu]^\top \in \mathbb{Z}_q^m$. Set $\boldsymbol{\beta}_0 = \mathbf{A}^\top\boldsymbol{s} + \boldsymbol{e}$, $\boldsymbol{\beta}_1 = \mathbf{P}^\top\boldsymbol{s} + \boldsymbol{e}' + \boldsymbol{b}$.
3. For $i \in [\ell]$, sample $\mathbf{R}_i \xleftarrow{\$} \{-1,1\}^{m\times m}$ and set $\boldsymbol{u}_i = (\mathbf{A}_i + y_i \cdot \mathbf{G})^\top\boldsymbol{s} + \mathbf{R}_i^\top\boldsymbol{e}$.
4. For $j \in [t]$, sample $\mathbf{R}'_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ and set $\boldsymbol{v}_j = (\mathbf{B}_j + x_j \cdot \mathbf{G})^\top\boldsymbol{s} + (\mathbf{R}'_j)^\top\boldsymbol{e}$.
5. Output the ciphertext $\mathsf{ct}_{\boldsymbol{y}} = \big(\boldsymbol{y}, \boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}\big)$.

$\mathsf{PH.Dec}(\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma}, \mathsf{ct}_{\boldsymbol{y}})$: Given as input a secret key and a ciphertext, the algorithm does the following steps:

1. Compute $\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_\mathsf{ct}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C} \circ \mathsf{IP}, \boldsymbol{y}\big)$.
2. Compute $\boldsymbol{\eta} = \boldsymbol{\beta}_1 - \mathbf{K}^\top \begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}\end{pmatrix}$.
3. Round each coordinate of $\boldsymbol{\eta}$. If $[\mathsf{Round}(\boldsymbol{\eta}[1]), \cdots, \mathsf{Round}(\boldsymbol{\eta}[m-1])] = \mathbf{0}$ then set $\mu = \mathsf{Round}(\boldsymbol{\eta}[m])$ and output μ. Otherwise, output $\perp$.

Theorem 5.3 *Assuming the hardness of* LWE*, then the scheme described in Section 5.1 is a* PHPE *for the class $\mathcal{F}$, achieving* $(1, \mathsf{poly})$-Sel-Sim *security that allows at most one* 1*-key pre-challenge query (and an unbounded polynomial number of* 0*-keys for both pre and post-challenge queries), according to Definition 2.17.*

For clarity of the presentation, we defer the correctness, parameter setting and the detailed proof of Theorem 5.3 to the supplementary material in Sections A.1, A.2, and A.3.

5.2 (Q, poly)-Partially Hiding Predicate Encryption

In this section, we upgrade our basic scheme to handle arbitrary pre- and post-challenge 1-key queries up to Q times (and any unbounded polynomially many 0-keys). Our upgrading technique is similar to that of Agrawal [2] (the Q-bounded PHPE) except that (1) we adopt our new sampling procedure in Section 4 for the key generation, (2) we use a simple secret sharing encoding over the message in a novel way, and (3) we take a more efficient way to generate cover-free sets by using a technique of [11]. Our resulting scheme achieves (Q, poly) simulation-based selective security with ciphertext growth additively with $O(Q)$, allowing general 1-key queries up to Q times, whereas the prior scheme of Agrawal [2] requires the adversary to be committed to all the functions of the 1-key queries right after seeing the public parameters, and the ciphertext size grows additively with $O(Q^2)$.

Before presenting the theorem, we first define the following set sampling algorithm.

Lemma 5.4 *Let $N = Qv\kappa^2$ and $v = \Theta(\kappa)$. There exists an efficient sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$ with the following properties: (1) The sampler always outputs a set $\Delta \subset [N]$ with cardinality v; (2) For independent samples $\Delta_1, \ldots, \Delta_Q$ from $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, the sets are cover-free with probability $(1 - 2^{-\Omega(\kappa)})$, i.e., for all $i \in [Q]$, $\mathsf{Pr}\left[\Delta_i \backslash \left(\bigcup_{j \neq i} \Delta_j\right) \neq \phi\right] \geq 1 - 2Q \cdot 2^{-\Omega(\kappa)}$.*

Proof. We construct $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$ as follows.

- The sampler first defines an (arbitrary) bijection $h : [N] \to [Q] \times [v\kappa^2]$.
- The sampler selects $i \in [Q]$ uniformly random, and a random $\Delta' \subset [v\kappa^2]$ of cardinality v.
- The sampler sets $\Delta = \{h^{-1}(i, j) : j \in \Delta'\}$, and outputs Δ.

The analysis of $\mathsf{Sampler}_{\mathsf{Set}}$ is similar to that in [11], so we just sketch the proof idea. We first observe that the bijection splits $[N]$ into Q buckets, each with $v\kappa^2$ elements. If we randomly throw Q balls to the buckets, then from the Chernoff bound, we have with at least probability $(1 - Q \cdot 2^{-\Omega(\kappa)})$ that all buckets will contain at most κ balls. These buckets correspond to the first index i. Suppose each bucket contains at most κ balls, where each ball corresponds to a random subset in the second index. Then by Lemma 2.2, for certain bucket, the probability that κ random subsets of size v are cover-free is at least $(1 - 2^{-\Omega(\kappa)})$. Furthermore, by union bound, we know that the independent samples $\Delta_1, \ldots, \Delta_Q$ from $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$ are cover free with at least probability $(1 - Q \cdot 2^{-\Omega(\kappa)})$.

The proof of this lemma simply follows from these two facts. □

In general we can choose κ to be $\omega(\log\lambda)$ to achieve $\mathsf{negl}(\lambda)$ security in the asymptotic setting, or say $\lambda^{1/3}$ to achieve $2^{-\Omega(\lambda)}$ security in the concrete setting.

Below we present the construction.

$\mathsf{QPH.Setup}(1^\lambda, 1^t, 1^\ell, 1^d, 1^Q)$: Given as input the security parameter λ, the length of the private and public attributes, t and ℓ respectively, the depth of the circuit family d, and Q as the upper bound of 1-key queries, do the following:

1. Choose public parameters (q, ρ, s, N, v) as described in the following parameter setting paragraph.
2. Choose random matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ for $i \in [\ell], \mathbf{B}_j \in \mathbb{Z}_q^{n\times m}$ for $j \in [t]$, and $\mathbf{P}_k \in \mathbb{Z}_q^{n\times m}$ for $k \in [N]$.
3. Sample $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, q, m)$.
4. Output the public and master secret keys.

$$\mathsf{PH.mpk} = (\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \mathbf{A}, \{\mathbf{P}_k\}_{k\in[N]}), \mathsf{PH.msk} = (\mathbf{T}_\mathbf{A})$$

$\mathsf{QPH.KeyGen}(\mathsf{PH.msk}, \widehat{C}\circ\mathsf{IP}_\gamma)$: Given as input a circuit description $\widehat{C}\circ\mathsf{IP}_\gamma$ and the master secret key, do the following:

1. Let $\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{pk}}(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \widehat{C}\circ\mathsf{IP})$.
2. Sample a random subset $\Delta \subset [N]$ according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$ with $|\Delta| = v$, and compute the subset sum $\mathbf{P}_\Delta = \sum_{k\in\Delta}\mathbf{P}_k$.
3. Sample matrix $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, and let $\mathbf{U} = \mathbf{P}_\Delta - \mathbf{A}\mathbf{J}$.
4. Sample $\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\mathbf{G}, \mathbf{T}_\mathbf{A}, \mathbf{U}, s)$ for Gaussian parameter s, i.e., the equation holds for $[\mathbf{A}|\mathbf{A}_{\widehat{C}\circ\mathsf{IP}}+\gamma\mathbf{G}]\cdot\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{U} \mod q$.
5. Let $\mathbf{K} = \begin{bmatrix}\mathbf{J}+\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix}$, and output $\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma} = (\Delta, \mathbf{K})$.

$\mathsf{QPH.Enc}(\mathsf{PH.mpk}, (\boldsymbol{x}, \boldsymbol{y}), \mu)$: Given as input the master public key, the private attributes $\boldsymbol{x}$, public attributes $\boldsymbol{y}$ and message μ, do the following:

1. Sample $\boldsymbol{s} \leftarrow \mathcal{D}_{\mathbb{Z}^n,s_B}$ and error terms $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_B}$ and $\boldsymbol{e}'_k \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}$ for $k \in [N]$.
2. Set $\boldsymbol{\beta}_0 = \mathbf{A}^\top\boldsymbol{s}+\boldsymbol{e}$, $\boldsymbol{b}_k = [0, \cdots, 0, \frac{\lceil q/2\rceil}{v}\mu] \in \mathbb{Z}_q^m$ for $k \in [N]$, and compute the following vectors as: $\{\boldsymbol{\beta}_{1,k} = \mathbf{P}_k^\top\boldsymbol{s} + \boldsymbol{e}'_k + \boldsymbol{b}_k\}_{k\in[N]}$.
3. For $i \in [\ell]$, sample $\mathbf{R}_i \xleftarrow{\$} \{-1,1\}^{m\times m}$ and set $\boldsymbol{u}_i = (\mathbf{A}_i+y_i\cdot\mathbf{G})^\top\boldsymbol{s}+\mathbf{R}_i^\top\boldsymbol{e}$.
4. For $j \in [t]$, sample $\mathbf{R}'_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ and set $\boldsymbol{v}_j = (\mathbf{B}_j + x_j\cdot\mathbf{G})^\top\boldsymbol{s} + (\mathbf{R}'_j)^\top\boldsymbol{e}$.
5. Output the ciphertext $\mathsf{ct}_{\boldsymbol{y}} = \big(\boldsymbol{y}, \boldsymbol{\beta}_0, \{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}, \{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}\big)$.

$\mathsf{QPH.Dec}(\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma}, \mathsf{ct}_{\boldsymbol{y}})$: Given as input a secret key $\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma} := (\Delta, \mathbf{K})$ and a ciphertext, do the following:

1. Compute $\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C}\circ\mathsf{IP}, \boldsymbol{y}\big)$.
2. Compute $\boldsymbol{\eta} = \sum_{k\in\Delta}\boldsymbol{\beta}_{1,k} - \mathbf{K}^\top\begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}\end{pmatrix}$.
3. Round each coordinate of $\boldsymbol{\eta}$. If $[\mathsf{Round}(\boldsymbol{\eta}[1]), \cdots, \mathsf{Round}(\boldsymbol{\eta}[m-1])] = \mathbf{0}$ then set $\mu = \mathsf{Round}(\boldsymbol{\eta}[m])$ and output μ. Otherwise, output $\perp$.

Theorem 5.5 *Assuming the hardness of* LWE*, then the* QPHPE *scheme described in Section 5.2 is* (Q, poly)-Sel-Sim *secure that allows both pre- and post-challenge* 1*-key queries up to* Q *times and* 0*-key queries for an unbounded polynomial times, as Definition 2.17.*

For clarity of the presentation, we defer the correctness and parameter setting to the supplementary material in Sections B.1 and B.2, respectively. Additionally, we just describe the simulator Sim for Theorem 5.5 here, and defer the detailed proof to Section B.3.

Simulator $\mathsf{Sim}(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|}, b, \mathsf{st})$:

1. $\mathsf{Sim}_1(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all public parameters as in the real PH.Setup, except that it runs $(\mathbf{A}', \mathbf{T}_{\mathbf{A}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$, then parses $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \boldsymbol{z}^\top \end{bmatrix}$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and sets $\mathbf{A}$ be the public matrix in PH.mpk.
2. $\mathsf{Sim}_2(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all keys using the real PH.KeyGen.
3. $\mathsf{Sim}_3(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|}, b, \mathsf{List})$: It takes as input the public attributes $\boldsymbol{y}$, the size of the private attributes $\boldsymbol{x}$, the message b, and a list List. It constructs the challenge ciphertext as follows.
   - It samples $\boldsymbol{u}_i, \boldsymbol{v}_j$ independently and uniformly from $\mathbb{Z}_q^m$, and sets $\boldsymbol{\beta}_0 = \boldsymbol{z}$, where $\boldsymbol{z}$ is the vector prepared in Sim_1.
   - If $(b, \mathsf{List}) = \perp$, it computes $\{\boldsymbol{\beta}_{1,k}\}_{k \in [N]}$ as follows:
     - Sample random vectors $\widetilde{\boldsymbol{\beta}}_k$ from $\mathbb{Z}_q^m$ for $k \in [N]$.
     - Choose $2Q$ random subsets $\Delta_1, \cdots, \Delta_Q, \Delta'_1, \cdots, \Delta'_Q$ of $[N]$ according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, each of which has cardinality v. Note that with an overwhelming probability, the $2Q$ subsets would be cover-free under our parameter selection.
     - Generate random shares $\{b_k\}_{k \in [N]}$ over $\mathbb{Z}_q$ under the following constraints: for $\hat{i} \in [Q]$, (1) $\sum_{k \in \Delta_{\hat{i}}} b_k = 0$, and (2) $\sum_{k \in \Delta'_{\hat{i}}} b_k = \lceil q/2 \rceil$. This can be done efficiently by the cover-freeness of the subsets, using the following standard procedure.
       First, let $\delta_{\hat{i}}$ be a unique index that only appears in $\Delta_{\hat{i}}$ but not the other subsets, and $\delta'_{\hat{i}}$ be a unique index of $\Delta'_{\hat{i}}$. To generate the random shares $\{b_k\}_{k \in [N]}$, we first sample b_k randomly for all $k \in [N] \setminus (\{\delta_{\hat{i}}\}_{\hat{i} \in [Q]} \cup \{\delta'_{\hat{i}}\}_{\hat{i} \in [Q]})$, and then fix $b_{\delta_{\hat{i}}} = -\sum_{k \in \Delta_{\hat{i}} \setminus \{\delta_{\hat{i}}\}} b_k$ for $\hat{i} \in [Q]$, and similarly $b_{\delta'_{\hat{i}}} = \lceil q/2 \rceil - \sum_{k \in \Delta'_{\hat{i}} \setminus \{\delta'_{\hat{i}}\}} b_k$ for $\hat{i} \in [Q]$.
     - Set $\boldsymbol{b}_k = [0, \cdots, 0, b_k] \in \mathbb{Z}_q^m$ for $k \in [N]$, and sample errors $\{\boldsymbol{e}'_k\}_{k \in [N]}$ from the distribution $\mathcal{D}_{\mathbb{Z}_q^m, s_D}$.
     - Set $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}'_k$ for $k \in [N]$.
   - If $b = \mu$ and $\mathsf{List} = \{\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}\}_{\hat{i} \in [Q']}$ for some $Q' \leq Q$, it computes the simulated ciphertext as follows.

- For $\hat{i} \in [Q']$, compute $\boldsymbol{u}_{\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP}, \boldsymbol{y}\big)$, and let $\left(\Delta_{\hat{i}}, \mathbf{K}^*_{\hat{i}} = \begin{bmatrix} \mathbf{J}^*_{\hat{i}} + \mathbf{K}^*_{\hat{i},1} \\ \mathbf{K}^*_{\hat{i},2} \end{bmatrix}\right)$ be the keys for $\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}$, generated by Sim_2 for the pre-challenge 1-key queries.
- Sample $Q - Q'$ random subsets of cardinality v according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, i.e., $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q'+1,Q]}$, starting with the index $Q'+1$ and ending with Q. We know that by our setting of parameters, the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ are cover-free with an overwhelming probability.
- Compute vectors $\{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}$ as follows:
  - Sample random shares $\{\mu_k\}_{k\in[N]}$ conditioned that $\sum_{k\in\Delta_{\hat{i}}} \mu_k = \lceil q/2 \rceil \mu$ for $\hat{i} \in [Q]$. Then set $\boldsymbol{b}_k = [0, \cdots, 0, \mu_k]$ for $k \in [N]$.
  - Sample random vectors $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ condition on the following equations:

$$\sum_{k\in\Delta_{\hat{i}}} \widetilde{\boldsymbol{\beta}}_k = \begin{bmatrix} \mathbf{J}^*_{\hat{i}} + \mathbf{K}^*_{\hat{i},1} \\ \mathbf{K}^*_{\hat{i},2} \end{bmatrix}^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}} \end{pmatrix} \text{ for } \hat{i} \in [Q'].$$

    The above two steps can be done efficiently due to the cover-freeness of the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$. The procedure is the same as we have presented in the previous case.
  - Sample errors $\{\boldsymbol{e}_k\}_{k\in[N]}$ according $\mathcal{D}_{\mathbb{Z}_q^m, s_D}$.
  - Set $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}'_k$ for $k \in [N]$.

– It outputs the challenge ciphertext

$$\mathsf{ct}^* = \big(\{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}, \boldsymbol{y}, \boldsymbol{\beta}_0, \{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}\big).$$

4. $\mathsf{Sim}_4(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: If the query is a 0-key, then it generates the key using the real $\mathsf{QPH.KeyGen}$. Otherwise, we denote function $\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}$ be the adversary's 1-key query and $(\mu, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}})$ be the message received from the oracle $\mathcal{O}$. Here we use index $\hat{i} \in [Q]$ to denote the number of overall 1-key queries up to this point. Then the simulator computes as follows.
   – The simulator first considers the following two cases to determine the parameter Δ:
     - Case 1: $Q' = 0$, i.e., the adversary did not make any 1-key pre-challenge query.
       * If $\mu = 0$, set $\Delta := \Delta_{\hat{i}}$.
       * Else $\Delta := \Delta'_{\hat{i}}$, where $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ and $\{\Delta'_{\hat{i}}\}_{\hat{i}\in[Q]}$ are the subsets prepared by Sim_3 in the previous procedure.
     - Case 2: $1 \le Q' < Q$, i.e., the adversary had made Q' 1-key pre-challenge queries.
       * Set $\Delta := \Delta_{\hat{i}}$ where $\Delta_{\hat{i}}$ is the subset prepared by Sim_3 (where μ had been received by Sim_3) in the previous procedure.
   – Compute $\mathbf{P}^*_\Delta = \sum_{k\in\Delta} \mathbf{P}_k$, and compute $\widetilde{\boldsymbol{\beta}}_\Delta = \sum_{k\in\Delta} \widetilde{\boldsymbol{\beta}}_k$, where $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ are the vectors prepared by Sim_3 in the previous procedure.

– Compute $\mathbf{A}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{pk}}(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP})$, and compute $\boldsymbol{u}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP}, \boldsymbol{y}\big)$.

– Sample $\mathbf{J}^*_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, and use $\mathbf{T}_{\mathbf{A}'}$ to sample $\begin{bmatrix}\mathbf{K}^*_{\hat{i},1}\\ \mathbf{K}^*_{\hat{i},2}\end{bmatrix} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times m},s}$ such that

$$\begin{bmatrix}\mathbf{A} & \mathbf{A}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}}\\ \boldsymbol{\beta}_0^\top & \boldsymbol{u}^\top_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{K}^*_{\hat{i},1}\\ \mathbf{K}^*_{\hat{i},2}\end{bmatrix} = -\begin{bmatrix}\mathbf{A}\\ \boldsymbol{\beta}_0^\top\end{bmatrix} \cdot \mathbf{J}^*_{\hat{i}} + \begin{bmatrix}\mathbf{P}^*_{\Delta}\\ \widetilde{\boldsymbol{\beta}}^\top_{\Delta}\end{bmatrix}.$$

– Output $\mathsf{sk}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}_{\gamma_{\hat{i}}}} = \Big(\Delta, \begin{bmatrix}\mathbf{J}^*_{\hat{i}} + \mathbf{K}^*_{\hat{i},1}\\ \mathbf{K}^*_{\hat{i},2}\end{bmatrix}\Big)$.

5.3 Semi-Adaptively Secure Partially Hiding Predicate Encryption

In this section, we show how to upgrade our PHPE in Section 5.2 from (Q, poly)-Sel-SIM security to (Q, poly)-SA-SIM security. Technically, we follow the idea of [21], yet in the case of bounded-length attributes (as used in this work). Below, we present the detailed construction.

Let $\mathsf{PH}_{\mathsf{Sel}} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ be a PHPE with private-public attribute space $\{0,1\}^t \times \{0,1\}^\ell$, message space $\mathcal{M}$, and function class $\mathcal{F}$ that is closed under bit-shift on $\{0,1\}^t \times \{0,1\}^\ell$ (i.e., for any $f \in \mathcal{F}$, $(\boldsymbol{r}, \boldsymbol{r}') \in \{0,1\}^t \times \{0,1\}^\ell$, we have $f_{\boldsymbol{r},\boldsymbol{r}'}(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}\oplus\boldsymbol{r}, \boldsymbol{y}\oplus\boldsymbol{r}') \in \mathcal{F}$). Moreover, the encryption algorithm $\mathsf{Enc}((\boldsymbol{x}, \boldsymbol{y}), \mu)$ can be decomposed into three parts: $\mathsf{Enc}_1(\mu; R)$, $\{\mathsf{Enc}_2(x_i; R)\}_{i\in[t]}$, $\{\mathsf{Enc}_3(y_i; R)\}_{i\in[\ell]}$, where R is the common random string among the three algorithms, x_i is the i-th bit of the attribute $\boldsymbol{x}$ whose bit-length is ℓ, and similarly y_i is the i-th bit of $\boldsymbol{y}$. Intuitively, the encryption procedure is done by three different components: with a common random string R, Enc_1 encodes the message, and both Enc_2 and Enc_3 encode the private/public attributes in the bit-by-bit manner.

Additionally, let $\mathsf{PKE} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}\}$ be any semantically secure public-key encryption. Then our transformation is defined as below.

$\mathsf{PH}_{\mathsf{SA}}.\mathsf{Setup}(1^\lambda, 1^t, 1^\ell)$: the algorithm takes the following steps:
- Run the underlying setup $(\mathsf{mpk}_{\mathsf{Sel}}, \mathsf{msk}_{\mathsf{Sel}}) \leftarrow \mathsf{PH}_{\mathsf{Sel}}.\mathsf{Setup}(1^\lambda, 1^\ell)$.
- Generate $\{\mathsf{PKE}.\mathsf{pk}_{i,b}, \mathsf{PKE}.\mathsf{sk}_{i,b}\}_{i\in[t],b\in\{0,1\}}$, $\{\mathsf{PKE}.\mathsf{pk}'_{i,b}, \mathsf{PKE}.\mathsf{sk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ from the scheme PKE.
- Sample a random string $(\boldsymbol{r}, \boldsymbol{r}') \in \{0,1\}^t \times \{0,1\}^\ell$.
- Finally output $\mathsf{mpk}_{\mathsf{SA}} = (\mathsf{mpk}_{\mathsf{Sel}}, \{\mathsf{PKE}.\mathsf{pk}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{PKE}.\mathsf{pk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$ as the master public key, and keep private $\mathsf{msk}_{\mathsf{SA}} = (\mathsf{msk}_{\mathsf{Sel}}, \{\mathsf{PKE}.\mathsf{sk}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{PKE}.\mathsf{sk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \boldsymbol{r}, \boldsymbol{r}')$ as the master secret key.

Note: Here Setup might implicitly take input $1^d, 1^Q$ for circuit depth and an upper bound of the 1-key queries. For simplicity, we omit the description.

$\mathsf{PH}_{\mathsf{SA}}.\mathsf{KeyGen}(\mathsf{msk}_{\mathsf{SA}}, f \in \mathcal{F})$: the algorithm defines a related function $f_{\boldsymbol{r},\boldsymbol{r}'}(\boldsymbol{x}, \boldsymbol{y}) := f(\boldsymbol{x} \oplus \boldsymbol{r}, \boldsymbol{y} \oplus \boldsymbol{r}')$, and runs $\mathsf{sk}_{\mathsf{Sel},f} \leftarrow \mathsf{PH}_{\mathsf{Sel}}(\mathsf{msk}_{\mathsf{Sel}}, f_{\boldsymbol{r},\boldsymbol{r}'})$. Then it returns $(\boldsymbol{r}, \boldsymbol{r}', \{\mathsf{PKE}.\mathsf{sk}_{i,r_i}\}_{i\in[t]}, \{\mathsf{PKE}.\mathsf{sk}'_{i,r'_i}\}_{i\in[\ell]}, \mathsf{sk}_{\mathsf{Sel},f})$ as the secret key.

$\mathsf{PH}_{\mathsf{SA}}.\mathsf{Enc}(\mathsf{mpk}_{\mathsf{SA}}, (\boldsymbol{x}, \boldsymbol{y}), \mu)$: the algorithms runs the following steps:

- Sample a random string R.
- Run $\mathsf{ct}_1 \leftarrow \mathsf{PH}_{\mathsf{Sel}}.\mathsf{Enc}_1(\mu; R)$, $\{L_{i,b} \leftarrow \mathsf{PH}_{\mathsf{Sel}}.\mathsf{Enc}_2(x_i \oplus b; R)\}_{i\in[t],b\in\{0,1\}}$, and $\{L'_{i,b} \leftarrow \mathsf{PH}_{\mathsf{Sel}}.\mathsf{Enc}_3(y_i \oplus b; R)\}_{i\in[\ell],b\in\{0,1\}}$.
- Generate $\{\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{PKE}.\mathsf{pk}_{i,b}, L_{i,b})\}_{i\in[t],b\in\{0,1\}}$ and $\{\mathsf{ct}'_{i,b} \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{PKE}.\mathsf{pk}'_{i,b}, L'_{i,b})\}_{i\in[\ell],b\in\{0,1\}}$.
- Finally, output the ciphertext as $\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{ct}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$.

$\mathsf{PH}_{\mathsf{SA}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{SA},f}, \boldsymbol{y}, \mathsf{ct})$:the algorithm runs the following steps:

- Parse $\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{ct}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$.
- Run the PKE decryption on $\{\mathsf{ct}_{i,r_i}\}_{i\in[t]}$ and $\{\mathsf{ct}'_{i,r'_i}\}_{i\in[\ell]}$. Then obtain $\{L_{i,r_i}\}_{i\in[t]}$ and $\{L'_{i,r'_i}\}_{i\in[\ell]}$.
- View $(\mathsf{ct}_1, \{L_{i,r_i}\}_{i\in[t]}, \{L'_{i,r'_i}\}_{i\in[\ell]})$ as the ciphertext of $\mathsf{PH}_{\mathsf{Sel}}$, and decrypt it with $\mathsf{sk}_{\mathsf{Sel},f}$. Output the decrypted outcome.

Theorem 5.6 *Assume that* PKE *is semantically secure, and* $\mathsf{PH}_{\mathsf{Sel}}$ *is* (q_1, q_2)-Sel-SIM *secure for private-public attribute space* $\{0,1\}^t \times \{0,1\}^\ell$*, message space* $\mathcal{M}$*, and function class* $\mathcal{F}$ *that is closed under bit-shift on* $\{0,1\}^t \times \{0,1\}^\ell$*. Then the scheme* $\mathsf{PH}_{\mathsf{SA}}$ *is* (q_1, q_2)-SA-SIM *secure for the same attribute and message spaces and the function class* $\mathcal{F}$*.*

For clarity of the presentation, we defer the correctness and the proof of Theorem 5.6 to the supplementary material in Sections C.1, C.2, respectively.

6 (Q, poly)-SIM-secure Functional Encryption

In this section, we present the technique from [2], showing that a (Q, poly)-SIM-secure QPHPE with a fully homomorphic encryption scheme implies a (Q, poly)-SIM-secure FE, which is what we desire. In particular, we present a (Q, poly)-functional encryption scheme by bootstrapping our QPHPE scheme. Our construction $\mathit{\Pi}_{\mathsf{FE}} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ uses two components:

- a fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$;
- a partially-hiding predicate encryption scheme $\mathsf{QPHPE} = (\mathsf{QPH.Setup}, \mathsf{QPH.KeyGen}, \mathsf{QPH.Enc}, \mathsf{QPH.Dec})$ that achieves (Q, poly)-SA-SIM security.

Formally, we require a QPHPE scheme for the circuit family $\mathcal{C}$ where $\widehat{C} \circ \mathsf{IP} \in \mathcal{C}$ is defined as follows. Let the private attributes $\boldsymbol{x} = \boldsymbol{t}$ where $\boldsymbol{t}$ is the FHE secret, and public attributes $\boldsymbol{y} = (\widehat{\boldsymbol{a}}, \widehat{0}_1, \cdots, \widehat{0}_N)$, where $\widehat{\boldsymbol{a}}$ is an FHE encryption of vector $\boldsymbol{a}$, and $\widehat{0}_i$ is an FHE encryption of the bit 0 for $i \in [N]$, with large noise ψ. Then, define:

$$\widehat{C}\big(\{\widehat{0}_i\}_{i\in[N]}, \widehat{\boldsymbol{a}}\big) = \mathsf{FHE.Scale}\Big(\mathsf{FHE.Eval}(\widehat{\boldsymbol{a}}, C) + \sum_{j\in[N]} r_j \widehat{0}_j\Big)$$

$$\widehat{C} \circ \mathsf{IP}\big(\boldsymbol{t}, \{\widehat{0}_i\}_{i\in[N]}, \widehat{\boldsymbol{a}}\big) = \Big\langle \boldsymbol{t}, \widehat{C}\big(\{\widehat{0}_i\}_{i\in[N]}, \widehat{\boldsymbol{a}}\big)\Big\rangle \bmod q$$

$$\widehat{C} \circ \mathsf{IP}_\gamma\big(\boldsymbol{t}, \{\widehat{0}_i\}_{i\in[N]}, \widehat{\boldsymbol{a}}\big) = 1 \text{ iff } \widehat{C} \circ \mathsf{IP}\big(\boldsymbol{t}, \{\widehat{0}_i\}_{i\in[N]}, \widehat{\boldsymbol{a}}\big) = \gamma,\ 0 \text{ otherwise,}$$

where $\boldsymbol{r} = (r_1, \cdots, r_N) \in \{0,1\}^N$ is a parameter of the circuit $\widehat{C}$. For simplicity of notation, we often omit the expression of $\boldsymbol{r}$ and assume that it is part of the description of $\widehat{C}$. On the other hand, we require the noise ψ of encryption of 0 and the noise of evaluation of $\boldsymbol{a}$'s ciphertext satisfy that

$$\psi + \mathsf{Noise}\big(\mathsf{FHE.Eval}(\widehat{\boldsymbol{a}}, C)\big) \overset{s}{\approx} \psi. \tag{2}$$

Above, $\mathsf{FHE.Scale}$ is the modulus reduction algorithm as Theorem 2.20, taking input an FHE ciphertext over modulus p and reducing it to a ciphertext that is over modulo q, for a smaller yet perhaps still super-polynomial modulus q. Notice that this q is required to match the modulus of the used QPH, and the noise in the resulting ciphertext is required to be polynomially bounded. Below we describe how to achieve this.

We first note that q is required to be $\lambda^{\Omega(d)}$ from the QPH, where d is the depth of the circuit $\widehat{C}$. Therefore, q would be super-polynomial if the depth is beyond a constant, i.e., $O(1)$.[15] Next, it is without loss of generality to assume that the noise of the output of $\mathsf{FHE.Eval}$ is polynomially bounded, via the bootstrapping technique. Thus, the noise of $\mathsf{FHE.Eval}(\widehat{\boldsymbol{a}}, C) + \sum_{j\in[N]} r_j \widehat{0}_j$ can be dominated and upper bounded by the noise bound of ψ, say some super-polynomial B, with an overwhelming probability. Then by setting the FHE modulus as $p = qB + \mathsf{poly}(\lambda)$ for some small polynomial $\mathsf{poly}(\cdot)$, according to Theorem 2.20, the noise of the resulting FHE ciphertext after $\mathsf{FHE.Scale}$ would be polynomially bounded. This satisfies what we need for the construction.

For the sake of brevity, we slightly abuse notation and do not explicitly include the inputs (p, q) in the inputs to $\widehat{C}$.

6.1 Construction of (Q, poly) Functional Encryption

In this section, we present a private-index FE for the class $\mathcal{C} \times \{I\}$, where $\mathcal{C}$ contains all bounded-depth circuits.

$\mathsf{FE.Setup}(1^\lambda, 1^k, 1^d, 1^Q)$: The setup algorithm takes as input the security parameter λ, the index length k, the function depth d and the upper bound of 1-key queries Q, sets another useful parameter $N = O(Q)$, and then does the following:

1. Choose the FHE modulus p in which $\mathsf{FHE.Eval}(\cdot,\cdot)$ will be computed and the FHE modulus $q \in \mathsf{poly}(\lambda)$ in which decryption will be performed.
2. Invoke the setup algorithm for the QPHPE scheme for family $\mathcal{C}$ to get:

$$(\mathsf{QPH.mpk}, \mathsf{QPH.msk}) \leftarrow \mathsf{QPH.Setup}(1^\lambda, 1^t, 1^\ell, 1^{d'}, 1^Q),$$

where length of private attributes $t = |\mathsf{FHE.sk}|$, length of public attributes ℓ is the length of an FHE encryption of $(k+N)$ bits corresponding to the attributes $\boldsymbol{a}$ and 0, i.e. $\ell = (k+N) \cdot |\mathsf{FHE.ct}|$ and d' is the bound on the augmented FHE evaluation circuit.

[15] In fact, we can compute a polynomial-length branching program for any NC1 circuit within a polynomial modulus [36], yet this cannot be extended beyond the NC1 class.

3. Output $(\mathsf{FE.mpk} = (\mathsf{QPH.mpk}, N), \mathsf{FE.msk} = \mathsf{QPH.msk})$.

$\mathsf{FE.KeyGen}(\mathsf{FE.msk}, C)$: The key generation algorithm takes as input the master secret key $\mathsf{FE.msk}$ and a circuit C. It does the following:

1. Let $R \triangleq [\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B]$. For each $\gamma \in R$, sample a set $\Delta \leftarrow \mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$ (ref. Lemma 5.4 in Section 5.2), set $\boldsymbol{r} \in \{0,1\}^N$ as the indicator vector of Δ, and include $\boldsymbol{r}$ in the circuit description of $\widehat{C} \circ \mathsf{IP}_\gamma$ as described above.
2. For each $\gamma \in R$, compute
$$\mathsf{QPH.sk}_{\widehat{C} \circ \mathsf{IP}_\gamma} \leftarrow \mathsf{QPH.KeyGen}(\mathsf{QPH.msk}, \widehat{C} \circ \mathsf{IP}_\gamma).$$
3. Output the secret key as $\mathsf{sk}_C = \{\mathsf{QPH.sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}\}_{\gamma \in R}$.

$\mathsf{FE.Enc}(\mathsf{FE.mpk}, \boldsymbol{a}, \mu)$: The encryption algorithm does the following:

1. Sample a fresh FHE secret key $\mathsf{FHE.sk}$, and denote it by $\boldsymbol{t}$.
2. Compute an FHE encryption of $\boldsymbol{a}$ to get $\widehat{\boldsymbol{a}} = \mathsf{FHE.Enc}(\boldsymbol{t}, \boldsymbol{a})$.
3. Sample $\{\psi_i\}_{i \in [N]}$ to satisfy the formula (2) and compute N FHE encryptions of 0 with noise $\{\psi_i\}_{i \in [N]}$ as $\{\widehat{0}_i\}_{i \in [N]}$.
4. Set public attributes $\boldsymbol{y} = (\widehat{\boldsymbol{a}}, \widehat{0}_1, \cdots, \widehat{0}_N)$ and private attributes $\boldsymbol{x} = \boldsymbol{t}$.
5. Compute $\mathsf{QPH.ct}_{\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}_{i \in [N]}} = \mathsf{QPH.Enc}\big(\mathsf{QPH.mpk}, (\boldsymbol{x}, \boldsymbol{y}), \mu\big)$.
6. Output $\mathsf{ct}_{\boldsymbol{a}} = \big(\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}_{i \in [N]}, \mathsf{QPH.ct}_{\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}_{i \in [N]}}\big)$.

$\mathsf{FE.Dec}(\mathsf{sk}_C, \mathsf{ct}_{\boldsymbol{a}})$: Do the following:

1. Parse sk_C as the set $\{\mathsf{QPH.sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}\}_{\gamma \in R}$
2. For each $\gamma \in R$, let $\tau_\gamma = \mathsf{QPH.Dec}\big(\mathsf{ct}_{\boldsymbol{a}}, \mathsf{QPH.sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}\big)$. If there exists some value γ' for which $\tau_{\gamma'} \neq \perp$, then output $\mu = \tau_{\gamma'}$, else output $\perp$.

Correctness. Correctness follows from correctness of QPHPE. As we defined before

$$\widehat{C} \circ \mathsf{IP}_\gamma\big(\boldsymbol{t}, \{\widehat{0}_i\}_{i \in [N]}, \widehat{\boldsymbol{a}}\big) = 1 \text{ iff } \widehat{C} \circ \mathsf{IP}\big(\boldsymbol{t}, \{\widehat{0}_i\}_{i \in [N]}, \widehat{\boldsymbol{a}}\big) = \gamma,\ 0 \text{ otherwise.}$$

Note that $\widehat{C}\big(\{\widehat{0}_i\}_{i \in [N]}, \widehat{\boldsymbol{a}}\big) = \mathsf{FHE.Scale}\Big(\mathsf{FHE.Eval}(\widehat{\boldsymbol{a}}, C) + \sum_{j \in [N]} r_j \widehat{0}_j, q, p\Big)$ is a modulo p FHE ciphertext of message $C(\boldsymbol{a})$ by the properties of FHE. If $C(\boldsymbol{a}) = 1$, then for some $\theta \in \big[\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B\big]$, we have that:

$$\Big\langle \boldsymbol{t}, \widehat{C}\big(\{\widehat{0}_i\}_{i \in [N]}, \widehat{\boldsymbol{a}}\big) \Big\rangle = \theta \bmod q.$$

Therefore $\mathsf{QPH.Dec}\big(\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}, \mathsf{QPH.ct}_{\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}}, \widehat{C} \circ \mathsf{IP}_\gamma, \mathsf{QPH.sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}\big) = \mu$ iff $\gamma = \theta$. Otherwise, if $C(\boldsymbol{a}) = 0$, then θ must lie outside $\big[\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B\big]$. Hence $\mathsf{QPH.Dec}$ returns $\perp$ by correctness of QPHPE.

6.2 Security of (Q, poly) Functional Encryption in Section 6

Theorem 6.1 *Let $\mathcal{C}$ be the family of bounded depth circuits,* QPHPE *be a (Q, poly)-*SA-SIM *secure partially-hiding predicate encryption scheme for $\mathcal{F}$ as defined in Section 5, and* FHE *be a secure fully-homomorphic encryption scheme. Then the functional encryption scheme described above is (Q, poly)-*SA-SIM *secure for the class $\mathcal{C} \times \{I\}$.*

As the proof is similar to that of [2], we only describe the required simulator and sketch the hybrids.

Proof (Sketch). We construct a simulator FE.Sim as required by Definition 3.2.

Simulator $\mathsf{FE.Sim}(1^\lambda)$. The simulator is described as follow:

1. It invokes $\mathsf{QPHPE.Sim}_1(1^\lambda)$ to obtain the public parameters.
2. The adversary $\mathcal{A}$ outputs challenge index $\boldsymbol{a}$. $\mathsf{FE.Sim}_2$ does the following:
   - If $\mathcal{A}$ makes a 0-key query C, $\mathsf{FE.Sim}_2$ transforms C into $\{\widehat{C} \circ \mathsf{IP}_\gamma\}_{\gamma \in R}$, and sends this to $\mathsf{QPHPE.Sim}_2$. $\mathsf{FE.Sim}_2$ receives the secret key set $\{\mathsf{QPH.sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}\}_{\gamma \in R}$ from $\mathsf{QPHPE.Sim}_2$, and forwards them to $\mathcal{A}$.
   - If $\mathcal{A}$ makes a 1-key query C^*, $\mathsf{FE.Sim}_2$ samples γ_q such that formula (2) holds, and sets γ to denote its scaled down version modulo p, and then computes $\widehat{C}^* \circ \mathsf{IP}_\gamma$. For $\theta \in R \backslash \{\gamma\}$, $\mathsf{FE.Sim}_2$ computes $\{\widehat{C}^* \circ \mathsf{IP}_\theta\}$. Finally, $\mathsf{FE.Sim}_2$ sends $\{\widehat{C}^* \circ \mathsf{IP}_\gamma\}_{\gamma \in R}$ to $\mathsf{QPHPE.Sim}_2$, and forwards $\{\mathsf{QPH.sk}_{\widehat{C}^* \circ \mathsf{IP}_\gamma}\}_{\gamma \in R}$ to $\mathcal{A}$ upon receiving the secret key set from $\mathsf{QPHPE.Sim}_2$.
3. Upon receiving the tuple (b, List), $\mathsf{FE.Sim}_3$ samples an FHE secret key FHE.sk and sets $\widehat{\boldsymbol{a}} = \mathsf{FHE.Enc}(\mathsf{FHE.sk}, \mathbf{0})$ and $\widehat{0}_i = \mathsf{FHE.Enc}(\mathsf{FHE.sk}, 0)$ with noise ψ_i for $i \in [N]$. Then it invokes $\mathsf{QPHPE.Sim}_3\big((\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}_{i \in [N]}), 1^{|\mathsf{FHE.sk}|}, b, \mathsf{List}\big)$ to obtain QPH.ct. Finally, $\mathsf{FE.Sim}_3$ outputs $(\widehat{\boldsymbol{a}}, \{\widehat{0}_i\}_{i \in [N]}, \mathsf{QPH.ct})$.
4. $\mathsf{FE.Sim}_4$ is similar to $\mathsf{FE.Sim}_2$ except that it invokes $\mathsf{QPHPE.Sim}_4$ instead of $\mathsf{QPHPE.Sim}_2$ to generate secret keys.

The proof follows from a series of hybrids described below.

Hybrid 0: The real experiment.
Hybrid 1: In this hybrid, the FHE encryption $\widehat{\boldsymbol{a}}$ is generated honestly but the remainder of the experiment is simulated.
Hybrid 2: The simulated experiment.

This completes the sketch of the proof. □

We notice that the required QPHPE can be instantiated by Theorems 5.5 and 5.6. Thus, we obtain the following corollary to summarize the final result.

Corollary 6.2 *Assuming the hardness of* LWE *for a sub-exponential modulus-to-noise ratio. Then for any bounded polynomial $Q = \mathsf{poly}(\lambda)$, there exists a (Q, poly)-*SA-SIM *secure* FE *for the class $\mathcal{C} \times \{I\}$.*

7 Constructions of FE with Public index

In this section, we notice that our two-stage sampling technique in Section 4 can be further used to derived several new feasibilities of FE with public index. Particularly, we first construct an adaptively-secure public-index FE for IB-FEIP in the IND-based setting. Then, we construct a semi-adaptively-secure public-index FE for AB-FEIP in the IND-based setting. Both of constructions are in the standard model.

7.1 Construction of IB-FEIP

In this section, we present the first functional encryption with public index for the class $\mathcal{I} \times (\mathsf{IP})$ where $\mathcal{I}$ is the class of identity comparison functions, and IP is the class of inner product functions over $\mathbb{Z}$. The scheme achieves $(1, \mathsf{poly})$-adaptive security.

$\mathsf{IB\text{-}FEIP.Setup}(1^\lambda, \mathcal{I}, \mathsf{IP}, \mathcal{X}, \mathcal{M})$: Given as input the security parameter λ, the class $\mathcal{I}$ of identity comparison functions, the class IP of inner product function indexed by vectors in $\{0, \ldots, v-1\}^\ell$, the index (or identity) space $\mathcal{X} = \{0,1\}^{\bar{\ell}}$ and the message space $\mathcal{M} = \{0, \ldots, p-1\}^\ell$ with $\ell, \bar{\ell}, v, p \in \mathbb{N}$, the algorithm does the following:

1. Choose modulus q, parameters $n, m, k, \rho, s, \sigma, \tau$ as described in next paragraph for parameters.
2. Sample matrices $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $i \in [\bar{\ell}]$, $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{n \times \ell}$.
3. Sample $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$.
4. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{A}, \{\mathbf{A}_i\}_{i \in [\bar{\ell}]}, \mathbf{P}), \mathsf{msk} := \mathbf{T}_\mathbf{A}.$$

$\mathsf{IB\text{-}FEIP.KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{y})$: Given as input a master secret key msk, an identity $\mathsf{id} = (b_1, \ldots, b_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$, and one vector $\boldsymbol{y} \in \{0, \ldots, v-1\}^\ell$, the algorithm does the following:

1. Let $\mathbf{A}_\mathsf{id} = \mathbf{G} + \sum_{i=1}^{\bar{\ell}} (b_i \mathbf{A}_i) \in \mathbb{Z}_q^{n \times m}$.
2. Sample matrix $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \rho}$, and let $\mathbf{U} = \mathbf{P} - \mathbf{A}\mathbf{J} (\mathrm{mod} q)$.
3. Sample $\begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_\mathsf{id}, \mathbf{T}_\mathbf{A}, \mathbf{U}, s)$ for Gaussian parameter s, i.e., the equation holds for $[\mathbf{A} | \mathbf{A}_\mathsf{id}] \cdot \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} = \mathbf{U} (\mathrm{mod} q)$.
4. Let $\mathbf{K}_\mathsf{id} = \begin{bmatrix} \mathbf{J} + \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} \in \mathbb{Z}_q^{2m \times \ell}$, and output $(\boldsymbol{y}, \mathsf{sk}_{\mathsf{id}, \boldsymbol{y}} = \mathbf{K}_\mathsf{id} \cdot \boldsymbol{y})$.

$\mathsf{IB\text{-}FEIP.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x})$: Given as input a master public key, an identity $\mathsf{id} = (b_1, \ldots, b_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$, and a vector $\boldsymbol{x} \in \mathcal{M}$, the algorithm does the following:

1. Let $\mathbf{A}_\mathsf{id} = \mathbf{G} + \sum_{i=1}^{\bar{\ell}} (b_i \mathbf{A}_i) \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{H}_\mathsf{id} = [\mathbf{A} | \mathbf{A}_\mathsf{id}] \in \mathbb{Z}_q^{n \times 2m}$.
2. Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
3. Sample $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$ for $i \in [\bar{\ell}]$, and compute $\mathbf{R}_\mathsf{id} = \sum_{i=1}^{\bar{\ell}} (b_i \cdot \mathbf{R}_i)$.

4. Sample error terms $\boldsymbol{e}_{1,1}, \boldsymbol{e}_{1,2} \leftarrow \mathcal{D}_{\mathbb{Z}^m, 2\sigma\tau}$, $\boldsymbol{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^\ell, 2\sigma\tau+\sigma}$.
5. Let $\boldsymbol{e} = \begin{pmatrix} \boldsymbol{e}_{1,1} \\ \boldsymbol{e}_{1,2} \end{pmatrix}$. Compute $\mathsf{ct}_1 = \mathbf{H}_{\mathsf{id}}^\top \cdot \boldsymbol{s} + \boldsymbol{e}$, and $\mathsf{ct}_2 = \mathbf{P}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_2 + \lfloor \frac{q}{k} \rfloor \cdot \boldsymbol{x}$.
6. Output the ciphertext $(\mathsf{ct}_1, \mathsf{ct}_2)$.

$\mathsf{IB\text{-}FEIP.Dec}(\boldsymbol{y}, \mathsf{sk}_{\mathsf{id},\boldsymbol{y}}, \mathsf{ct}_1, \mathsf{ct}_2)$: Given as input a secret key $(\boldsymbol{y}, \mathsf{sk}_{\mathsf{id},\boldsymbol{y}})$ and a ciphertext $(\mathsf{ct}_1, \mathsf{ct}_2)$, the algorithm does the following:

1. Compute $\mu' = \boldsymbol{y}^\top \cdot \mathsf{ct}_2 - \mathsf{sk}_{\mathsf{id},\boldsymbol{y}}^\top \cdot \mathsf{ct}_1 \bmod q$.
2. Output the value $\mu \in \{0, \ldots, k-1\}$ that minimizes $|\lfloor \frac{q}{k} \cdot \mu - \mu' \rfloor|$.

Parameters.

Our scheme is an essential improvement of selective $\mathsf{IB\text{-}FEIP}$ in [1] with similar parameters. For completeness, we present the details below. For the plaintext vectors $\boldsymbol{x} \in \mathcal{M} = \{0, \ldots, p-1\}^\ell$, and the decryption key vectors $\boldsymbol{y} \in \{0, \ldots, v-1\}^\ell$, we set $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \{0, \ldots, k-1\}$ with $k = \ell v p$. Since our construction is essential a combination of LWE-based adaptively secure FEIP in [5] (which is referred as ALS in the following contexts) and IBE from [3], we set the parameters in the following way:

- For the security and correctness of ALS, we need to set the parameters n_{ALS}, q_{ALS}, σ_{ALS}, ρ_{ALS}, $m_{\mathsf{ALS}} = 2n_{\mathsf{ALS}} \log q_{\mathsf{ALS}}$, α_{ALS} as in Appendix D.1.
- For the security reduction to ALS, we need to set $\sigma = \sigma_{\mathsf{ALS}}$, $q = q_{\mathsf{ALS}}$, $n = n_{\mathsf{ALS}}$, $m = m_{\mathsf{ALS}}$.
- For the algorithms $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$, we need to set $s > m \cdot \bar{\ell}\omega(\sqrt{\log m})$.
- For our new two-stage sampling algorithms in Section 4, we need to set $s \geq \omega(\sqrt{\log m})$ and $\rho \geq s \cdot \bar{\ell} \cdot m \cdot \lambda^{\omega(1)}$.
- For the algorithm $\mathsf{TrapGen}$, we need to set $m > 6n \log q$.
- For the hardness of $\mathsf{LWE}_{n,q,\alpha}$, we need to set $\alpha = \alpha_{\mathsf{ALS}}$ and $\alpha q > 2\sqrt{n}$.
- For the algorithm ReRand in the process of noise rerandomization, we need to set $\sigma \geq \omega\log(2m+\ell)$, $\tau \geq \sqrt{1 + c^2(\rho^2+\sigma^2) \cdot (2\sqrt{m}+\sqrt{\ell})^2 + \bar{\ell}^2 m^2}$, since $s_1(\mathbf{I}) = 1$, $s_1(\mathbf{R}_{\mathsf{id}^*}) \leq \bar{\ell} s_1(\mathbf{R}_i) \leq \bar{\ell} m$, and $s_1(\mathbf{J}^*) \leq c \cdot (\sqrt{\rho^2+\sigma^2})(2\sqrt{m}+\sqrt{\ell})$ for $\mathbf{R}_i^* \xleftarrow{\$} \{0,1\}^{m \times m}$ and $\mathbf{J}^* \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \sqrt{\rho^2+\sigma^2}}$.
- For correctness, we need to set $q \geq 8k\sigma m\tau\rho + 2k\ell v(2\sigma\tau + \sigma)$.
  Hence, we choose to use ALS parameters, and modify them in the following way to satisfy these restrictions.
  - $m > 6n \log q$.
  - $q \geq 8k\sigma m\tau\rho + 2k\ell v(2\sigma\tau + \sigma)$.
  - $s \geq m \cdot \bar{\ell}\omega(\sqrt{\log m})$.
  - $\rho \geq s \cdot \bar{\ell} \cdot m \cdot \lambda^{\omega(1)}$.
  - $\tau \geq \sqrt{1 + c^2(\rho^2+\sigma^2) \cdot (2\sqrt{m}+\sqrt{\ell})^2 + \bar{\ell}^2 m^2}$.

Lemma 7.1 (Correctness) *For parameters n, m, q, σ, ρ, s, τ, ℓ, p, v, k, chosen as in the previous paragraph, our scheme $\mathsf{IB\text{-}FEIP}$ is correct.*

This lemma can be easily verified as [1], so we omit the proof.

Theorem 7.2 (AD-IND Security) *Let λ be the security parameter and consider the functional encryption scheme* ALS *with parameters $q, \sigma, \rho, s, m, n, \alpha$ chosen as in the parameter setting paragraph of Section D.1. If* ALS *is adaptively secure under the* $\mathsf{LWE}_{q,\alpha,n}$ *assumption, then our scheme* IB-FEIP *with the above parameters setting is* AD-IND *secure under* $\mathsf{LWE}_{q,\alpha,n}$ *assumption.*

The proof follows the outline of the original adaptive IBE in [3]. The novelty of our proof is the use of our new two-stage sampling algorithm to support adaptive functional decryption key query.

Proof. Before proving this theorem, we first present some auxiliary algorithms.

Auxiliary Algorithms.

$\mathsf{IB\text{-}FEIP.Setup}_1(1^\lambda, \mathcal{I}, \mathsf{IP}, \mathcal{X}, \mathcal{M})$: Do the following:

1. Choose modulus q, parameters $n, m, k, \rho, s, \sigma, \tau$ as described in next paragraph for parameters.
2. Sample $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$.
3. Let $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - h_i \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $i \in [\bar{\ell}]$, where $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and $h_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [\bar{\ell}]$.
4. Sample $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{m \times \ell}$.
5. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{A}, \{\mathbf{A}_i\}_{i \in [\bar{\ell}]}, \mathbf{P}), \mathsf{msk} := (\mathbf{T_A}, \{\mathbf{R}_i\}_{i \in \bar{\ell}}).$$

$\mathsf{IB\text{-}FEIP.Setup}_2(1^\lambda, \mathcal{I}, \mathsf{IP}, \mathcal{X}, \mathcal{M})$: Do the following:

1. Choose modulus q, parameters $n, m, k, \rho, s, \sigma, \tau$ as described in next paragraph for parameters.
2. Sample $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$.
3. Let $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - h_i \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $i \in [\bar{\ell}]$, where $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and $h_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [\bar{\ell}]$.
4. Sample $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \sqrt{\rho^2 + s^2}}$, and set $\mathbf{P} = \mathbf{A} \cdot \mathbf{J} (\mathrm{mod} q)$.
5. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{A}, \{\mathbf{A}_i\}_{i \in [\bar{\ell}]}, \mathbf{P}), \mathsf{msk} := (\mathbf{T_A}, \{\mathbf{R}_i\}_{i \in \bar{\ell}}, \mathbf{J}).$$

$\mathsf{IB\text{-}FEIP.Setup}_3(1^\lambda, \mathcal{I}, \mathsf{IP}, \mathcal{X}, \mathcal{M})$: Do the following:

1. Choose modulus q, parameters $n, m, k, \rho, s, \sigma, \tau$ as described in next paragraph for parameters.
2. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.
3. Let $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - h_i \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $i \in [\bar{\ell}]$, where $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and $h_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [\bar{\ell}]$.
4. Sample $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \sqrt{\rho^2 + s^2}}$, and set $\mathbf{P} = \mathbf{A} \cdot \mathbf{J} (\mathrm{mod} q)$.

5. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{A}, \{\mathbf{A}_i\}_{i\in[\bar{\ell}]}, \mathbf{P}), \mathsf{msk} := (\{\mathbf{R}_i\}_{i\in\bar{\ell}}, \mathbf{J}).$$

$\mathsf{IB\text{-}FEIP.KeyGen}_1(\mathsf{PH.msk}, \mathsf{id}, \boldsymbol{y})$: Do the following:

1. Let $\mathbf{A}_{\mathsf{id}} = \mathbf{G} + \sum_{i=1}^{\bar{\ell}}(b_i\mathbf{A}_i) = \mathbf{A}\mathbf{R}_{\mathsf{id}} + h_{\mathsf{id}}\mathbf{G} \in \mathbb{Z}_q^{n\times m}$, where $\mathbf{R}_{\mathsf{id}} = \sum_{i=1}^{\bar{\ell}}(b_i\mathbf{R}_i) \in \mathbb{Z}_q^{m\times m}$ and $h_{\mathsf{id}} = 1 + \sum_{i=1}^{\bar{\ell}}(b_i h_i) \in \mathbb{Z}_q$.
2. For different types of key queries, we respond as follows:
   - 0-query $(\mathsf{id}, \boldsymbol{y})$ such that $\mathsf{id} \neq \mathsf{id}^*$. In this case, we have $h_{\mathsf{id}} \neq 0$. As the $\mathbf{G}$ does not vanish, the algorithm runs

     $$\mathbf{K}_{\mathsf{id}} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \mathbf{R}_{\mathsf{id}}, h_{\mathsf{id}}, \mathbf{P}, s),$$

     satisfying $[\mathbf{A}|\mathbf{A}\mathbf{R}_{\mathsf{id}} + h_{\mathsf{id}}\mathbf{G}]\mathbf{K}_{\mathsf{id}} = \mathbf{P} (\mathrm{mod} q)$.
   - 1-query $(\mathsf{id}, \boldsymbol{y})$ such that $\mathsf{id} = \mathsf{id}^*$. In this case, the algorithm samples $\mathbf{K}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times\ell}, s}$, and sets $\mathbf{K}_{\mathsf{id}} = \begin{bmatrix} \mathbf{J} - \mathbf{R}_{\mathsf{id}}\mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix}$.
   - Return $(\boldsymbol{y}, \mathbf{K}_{\mathsf{id}} \cdot \boldsymbol{y})$.

Now we establish the proof via the following hybrids, where the first one corresponds to the real world and security of the last one follows from the security of the ALS encryption scheme.

Hybrid 0: This is the real experiment with (m_0, m_1) as the challenge messages.
Hybrid 1: This hybrid is almost identical to Hybrid 0 except for replacing IB-FEIP.Setup with $\mathsf{IB\text{-}FEIP.Setup}_1$.
Hybrid 2: This hybrid is almost identical to Hybrid 1 except for replacing $\mathsf{IB\text{-}FEIP.Setup}_1$ with $\mathsf{IB\text{-}FEIP.Setup}_2$.
Hybrid 3: This hybrid is almost identical to Hybrid 2 except for replacing $\mathsf{IB\text{-}FEIP.Setup}_2$ with $\mathsf{IB\text{-}FEIP.Setup}_3$.
Hybrid 4: This hybrid is almost identical to Hybrid 3 except for replacing IB-FEIP.KeyGen with $\mathsf{IB\text{-}FEIP.KeyGen}_1$.

Next, for the last hybrid, we build a reduction from ALS, meaning that suppose the adversary has an noticeable advantage in Hybrid 4, then the reduction can break ALS with a similar advantage.

Particularly, upon receiving public keys $\mathbf{A}_{\mathsf{ALS}}$ and $\mathbf{D}_{\mathsf{ALS}}$ from the AD-CPA challenger $\mathcal{C}_{\mathsf{ALS}}$, the reduction simulates the view of the adversary $\mathcal{A}_{\mathsf{Hybrid}\ 4}$ in Hybrid 4 in the following way:

- Setup: The reduction sets $\mathbf{A} := \mathbf{A}_{\mathsf{ALS}}^{\top}$, and conducts the following steps:
  1. Choose $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times\ell}, \sqrt{\rho^2+s^2}}$, and compute $\mathbf{P} := \mathbf{D}_{\mathsf{ALS}}^{\top} + \mathbf{A}\mathbf{J}^* (\mathrm{mod} q)$,
  2. Choose $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m\times m}$ and $h_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [\bar{\ell}]$, and then compute $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - h_i \cdot \mathbf{G} \in \mathbb{Z}_q^{n\times m}$ for $i \in [\bar{\ell}]$.
  3. Send $(\mathbf{A}, \{\mathbf{A}_i\}_{i\in[\ell]}, \mathbf{P})$ as mpk to $\mathcal{A}_{\mathsf{Hybrid}\ 4}$.
- KeyGen for $\mathsf{id} \neq \mathsf{id}^*$: The reduction uses the 0-key query algorithm as in Hybrid 4.

– KeyGen for $(\mathsf{id}^*, \tilde{\boldsymbol{y}})$: The reduction asks $\mathcal{C}_{\mathsf{ALS}}$ for the functional decryption key with respect to $\tilde{\boldsymbol{y}}$, and then receives $\mathsf{sk}_{\tilde{\boldsymbol{y}}}$. Finally, the reduction computes and return $\mathsf{sk}_{\mathsf{id}^*,\tilde{\boldsymbol{y}}} = \begin{bmatrix} \mathbf{J}^* \cdot \tilde{\boldsymbol{y}} + \mathsf{sk}_{\tilde{\boldsymbol{y}}} - \mathbf{R}_{\mathsf{id}^*}\mathbf{K}_2\tilde{\boldsymbol{y}} \\ \mathbf{K}_2\tilde{\boldsymbol{y}} \end{bmatrix}$[16], where $\mathbf{K}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times\ell},s}$, $\mathbf{R}_{\mathsf{id}^*} = \sum_{i=1}^{\bar{\ell}}(b_i^*\mathbf{R}_i) \in \mathbb{Z}_q^{m\times m}$ with $\mathsf{id}^* = (b_1^*, \ldots, b_{\bar{\ell}}^*)$.
– Challenge ciphertext query for $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ from $\mathcal{A}_{\mathsf{Hybrid}\ 4}$: The reduction forwards $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ to $\mathcal{C}_{\mathsf{ALS}}$, and receives $(\mathsf{ct}_1^{\mathsf{ALS}}, \mathsf{ct}_2^{\mathsf{ALS}})$. Then, the reduction conducts the following steps:
  1. Run the algorithm ReRand in Lemma 2.10, i.e., $\mathsf{ReRand}([\mathbf{I}|\mathbf{R}_{\mathsf{id}^*}|\mathbf{J}^*], \mathsf{ct}_1^{\mathsf{ALS}}, \sigma, \tau)$, to obtain $(\mathsf{ct}_{1,1}^\top, \mathsf{ct}_{1,2}^\top, \mathsf{ct}_{1,3}^\top)^\top \in \mathbb{Z}_q^{2m+\ell}$.
  2. Set $\mathsf{ct}_1 = (\mathsf{ct}_{1,1}^\top, \mathsf{ct}_{1,2}^\top)^\top \in \mathbb{Z}_q^{2m}$, $\mathsf{ct}_2 = \mathsf{ct}_{1,3} + \mathsf{ct}_2^{\mathsf{ALS}} \in \mathbb{Z}_q^\ell$.
  3. Send $(\mathsf{ct}_1, \mathsf{ct}_2)$ to $\mathcal{A}_{\mathsf{Hybrid}\ 4}$.

Next, let us analyze above reduction process. Clearly, according to Lemma 2.10, for $\mathsf{ct}_1^{\mathsf{ALS}} = \mathbf{A}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_1^{\mathsf{ALS}} \in \mathbb{Z}_q^m$, we have $\mathsf{ct}_{1,1} = \mathbf{A}^\top \cdot \boldsymbol{s} + \boldsymbol{e}'_{1,1} \in \mathbb{Z}_q^m$, $\mathsf{ct}_{1,2} = (\mathbf{A}\mathbf{R}_{\mathsf{id}^*})^\top \cdot \boldsymbol{s} + \boldsymbol{e}'_{1,2} \in \mathbb{Z}_q^m$, $\mathsf{ct}_{1,3} = (\mathbf{A}\mathbf{J}^*)^\top \cdot \boldsymbol{s} + \boldsymbol{e}'_{1,3} \in \mathbb{Z}_q^\ell$, where $\boldsymbol{e}'_{1,1}, \boldsymbol{e}'_{1,2} \in \mathcal{D}_{\mathbb{Z}^m,2\sigma\tau}$, $\boldsymbol{e}'_{1,3} \in \mathcal{D}_{\mathbb{Z}^\ell,2\sigma\tau}$. Hence, it holds

$$\mathsf{ct}_1 = (\mathsf{ct}_{1,1}^\top, \mathsf{ct}_{1,2}^\top)^\top = [\mathbf{A}|\mathbf{A}\cdot\mathbf{R}_{\mathsf{id}^*}]^\top \cdot \boldsymbol{s} + \begin{pmatrix} \boldsymbol{e}'_1 \\ \boldsymbol{e}'_2 \end{pmatrix}.$$

Additionally, for $\mathsf{ct}_2^{\mathsf{ALS}} = \mathbf{D}_{\mathsf{ALS}} \cdot \boldsymbol{s} + \boldsymbol{e}_2^{\mathsf{ALS}} + \lfloor \frac{q}{k} \cdot \boldsymbol{x}_b \rfloor$ with $b \in \{0,1\}$, we have

$$\mathsf{ct}_2 = \mathsf{ct}_{1,3} + \mathsf{ct}_2^{\mathsf{ALS}} = (\mathbf{A}\cdot\mathbf{J}^* + \mathbf{P})^\top \cdot \boldsymbol{s} + \boldsymbol{e}'_2 + \lfloor \frac{q}{k} \cdot \boldsymbol{x}_b \rfloor,$$

where $\boldsymbol{e}'_2 = \boldsymbol{e}'_{1,3} + \boldsymbol{e}_2^{\mathsf{ALS}}$. For sufficiently large σ, σ', we know the distribution of $\boldsymbol{e}'_2$ is statistically close to $\mathcal{D}_{\mathbb{Z}^\ell,2\sigma\tau+\sigma}$.

Hence, the reduction faithfully simulate Hybrid 4, and thus turns an adversary with a non-negligible advantage in Hybrid 4 into an adversary that breaks ALS with a non-negligible advantage.

Finally, we sketch how to show indistinguishability between each two adjacent hybrids.

Lemma 7.3 Hybrid 0 *and* Hybrid 1 *are statistically indistinguishable.*

Proof. This lemma holds due to Lemma 2.16. We omit the detailed proof here. □

Lemma 7.4 Hybrid 1 *and* Hybrid 2 *are statistically indistinguishable.*

Proof. This lemma holds due to Lemma 4.1. We omit the detailed proof here. □

Lemma 7.5 Hybrid 2 *and* Hybrid 3 *are statistically indistinguishable.*

[16] Notice that the distributions of this secret keys are still statistically close to that of the direct usage of $\mathsf{Sampler}$-2, since $\mathbf{J}_i^*\tilde{\boldsymbol{y}} + \mathsf{sk}_{\tilde{\boldsymbol{y}}_i}^{(i)}$ and $\mathbf{J}_i^*\tilde{\boldsymbol{y}}$ are statistically close, due to noise smudging for our parameter setting.

Proof. This lemma holds due to Theorem 2.13. We omit the detailed proof here. □

Lemma 7.6 Hybrid 3 *and* Hybrid 4 *are statistically indistinguishable.*

Proof. This lemma holds due to Theorem 4.2. We omit the detailed proof here, as it is quite similar to that of Lemma B.3. □

□

7.2 Construction of AB-FEIP Modulo p

In this section, we present the second functional encryption with public index for the class $\mathcal{C} \times (\mathsf{IP} \bmod p)$ where $\mathcal{C}$ is general circuits of depth at most d, and $\mathsf{IP} \bmod p$ is inner product modulo p. The scheme achieves (Q, poly)-selective security. The selective security can be upgraded to semi-adaptive security, using a semantically secure PKE scheme as in Section 5.3. For simplicity of presentation, we only describe the case of selective security in this section.

$\mathsf{AB\text{-}FEIP.Setup}(1^\lambda, 1^Q, \mathcal{C}, \mathsf{IP}, \mathcal{X}, \mathcal{M})$: Given as input the security parameter λ, the upper bound Q of 1-key queries, the class $\mathcal{C}$ of general d-depth circuits mapping $\{0,1\}^{\bar{\ell}}$ to $\{0,1\}$, the class IP of inner product function indexed by vectors in $\mathbb{Z}_p^\ell$, the index (or attribute) space $\mathcal{X} = \{0,1\}^{\bar{\ell}}$, and the message space $\mathcal{M} = \mathbb{Z}_p^\ell$, the algorithm does the following:

1. Choose modulus $q = p^e$, parameters $e, n, m, \rho, s, \sigma, N, t$ in a similar way as described in the parameter setting paragraph of Section 7.1.
2. Choose random matrices $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $i \in [\bar{\ell}], \mathbf{P}_j \xleftarrow{\$} \mathbb{Z}_q^{n \times \ell}$ for $j \in [N]$.
3. Sample $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$.
4. Output the public and master secret keys.

$$\mathsf{mpk} := (\mathbf{A}, \{\mathbf{A}_i\}_{i \in [\bar{\ell}]}, \{\mathbf{P}_j\}_{j \in [N]}), \mathsf{msk} := \mathbf{T}_\mathbf{A}.$$

$\mathsf{AB\text{-}FEIP.KeyGen}(\mathsf{msk}, C, \boldsymbol{y})$: Given as input a master secret key msk, a general predicate $C \in \mathcal{C}$, and one vector $\boldsymbol{y} \in \mathbb{Z}_p^\ell$, the algorithm does the following:

1. Let $\mathbf{A}_C = \mathsf{Eval}_\mathsf{pk}(\{\mathbf{A}_i\}_{i \in [\bar{\ell}]}, C)$.
2. Sample a random subset $\Delta \subset [N]$ with $|\Delta| = t$ according sampler $\mathsf{Sampler}_\mathsf{Set}(N, Q, t)$, and compute the subset sum $\mathbf{P}_\Delta = \sum_{k \in \Delta} \mathbf{P}_k$.
3. Sample matrix $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \rho}$, and let $\mathbf{U} = \mathbf{P}_\Delta - \mathbf{A}\mathbf{J} (\mathrm{mod} q)$.
4. Sample $\begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_C, \mathbf{T}_\mathbf{A}, \mathbf{U}, s)$ for Gaussian parameter s, i.e., the equation holds for $[\mathbf{A}|\mathbf{A}_C] \cdot \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix} = \mathbf{U} (\mathrm{mod} q)$.
5. Let $\mathbf{K}_C = \begin{bmatrix} \mathbf{J} + \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix}$, and output $(\Delta, \boldsymbol{y}, \mathsf{sk}_{C,\boldsymbol{y}} = \mathbf{K}_C \cdot \boldsymbol{y})$.

$\mathsf{AB\text{-}FEIP.Enc}(\mathsf{mpk}, \mathsf{x}, \boldsymbol{x})$: Given as input a master public key, an $\mathsf{x} = (\mathsf{x}_1, \ldots, \mathsf{x}_{\bar{\ell}}) \in \mathcal{X} = \{0,1\}^{\bar{\ell}}$, and a vector $\boldsymbol{x} \in \mathcal{M}$, the algorithm does the following:

1. Let $\mathbf{H}_{\mathsf{x}} = [\mathbf{A}|\mathsf{x}_1\mathbf{G} + \mathbf{A}_1| \cdots |\mathsf{x}_{\bar{\ell}}\mathbf{G} + \mathbf{A}_{\bar{\ell}}] \in \mathbb{Z}_q^{n \times (\bar{\ell}+1)m}$.
2. Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{R}_i \xleftarrow{\$} \{-1,1\}^{m \times m}$ for $i \in [\bar{\ell}]$, and error terms $\boldsymbol{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, and $\boldsymbol{e}_{2,i} \leftarrow \mathcal{D}_{\mathbb{Z}^\ell, 2\sigma\tau+\sigma}$ for $i \in [N]$.
3. Set an encoding of $\boldsymbol{x} \in \mathbb{Z}_p^\ell$ as $\{\boldsymbol{x}'_i\}_{i\in[N]}$, where $\boldsymbol{x}'_i = \frac{\lceil q/2 \rceil}{v}\boldsymbol{x}$.
4. Let $\boldsymbol{e} = [\mathbf{I}_m|\mathbf{R}_1| \cdots |\mathbf{R}_{\bar{\ell}}]^\top \cdot \boldsymbol{e}_1 \in \mathbb{Z}_q^{(\bar{\ell}+1)m}$.
5. Compute $\mathsf{ct}_1 = \mathbf{H}_{\mathsf{x}}^\top \cdot \boldsymbol{s} + \boldsymbol{e}$, and $\mathsf{ct}_{2,i} = \mathbf{P}_i^\top \cdot \boldsymbol{s} + \boldsymbol{e}_{2,i} + p^{e-1} \cdot \boldsymbol{x}'_i$ for $i \in [N]$
6. Output the ciphertext $(\mathsf{x}, \mathsf{ct}_1, \{\mathsf{ct}_{2,i}\}_{i\in[N]})$.

$\mathsf{AB\text{-}FEIP.Dec}(\Delta, \boldsymbol{y}, \mathsf{sk}_{C,\boldsymbol{y}}, \mathsf{x}, \mathsf{ct}_1, \{\mathsf{ct}_2\}_{i\in[N]})$: Given as input a secret key $(\Delta, \boldsymbol{y}, \mathsf{sk}_{C,\boldsymbol{y}})$ and a ciphertext $(\mathsf{x}, \mathsf{ct}_1, \{\mathsf{ct}_2\}_{i\in[N]})$, the algorithm does the following:

1. If $\mathsf{P}(\boldsymbol{x}) \neq 0$, output $\perp$. Otherwise, conduct the following steps.
2. Let $\mathsf{ct}_1 = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{1,\ell}) \in \mathbb{Z}_q^{(\ell+1)m}$.
3. Compute $\mathsf{ct}_{\mathsf{P}} = \mathsf{Eval}_{\mathsf{ct}}(\mathsf{P}, \{(\mathsf{x}_i, \mathbf{A}_i, \mathsf{ct}_{1,i})\}_{i=1}^{\ell}) \in \mathbb{Z}_q^m$.
4. Let $\mathsf{ct}'_{\mathsf{P}} = (\mathsf{ct}_{1,0}, \mathsf{ct}_{\mathsf{P}}) \in \mathbb{Z}_q^{2m}$.
5. Compute $u' = \boldsymbol{y}^\top \cdot \sum_{i\in\Delta}(\mathsf{ct}_{2,i}) - \mathsf{sk}_{\mathsf{P},\boldsymbol{y}}^\top \cdot \mathsf{ct}'_{\mathsf{P}} \bmod q$.
6. Output the value $\mu \in \mathbb{Z}_p$ that minimizes $\lfloor |p^{e-1} \cdot \mu - \mu'| \rfloor$.

Theorem 7.7 (Sel-IND Security) *Let λ be the security parameter and consider the functional encryption scheme N-ALS with parameters $q, \sigma, \rho, s, m, n, \alpha$ chosen as in the parameter setting paragraph of Section D.2. If N-ALS is secure under the $\mathsf{LWE}_{q,\alpha,n}$ assumption, then our scheme AB-FEIP is Sel-IND secure under $\mathsf{LWE}_{q,\alpha,n}$ assumption.*

The security of our construction relies on the improved ALS scheme, which we called N-ALS from the work [47]. We present the necessary backgrounds in Section D. Similar to proof of Theorem 7.2, this proof follows the outline of the original selective ABE in [12], except that we use our new two-stage sampling algorithm and the secret sharing technique in Section 5.2. To avoid repetition, we just provide a sketch of the proof.

Proof (Sketch). We prove the theorem via a series of hybrids. First, we view the real experiment as Hybrid_0. Then, in Hybrid 1, we replace the random matrices $\mathbf{A}_i$ with $\mathbf{A} \cdot \mathbf{R}_i - \mathsf{x}_i^*\mathbf{G} (\text{mod } q)$ for $i \in [\bar{\ell}]$, where $\mathbf{R}_i \xleftarrow{\$} \{-1,1\}^{m\times m}$ and $\mathsf{x}^* = (\mathsf{x}_1^*, \ldots, \mathsf{x}_{\bar{\ell}}^*)$ is the challenge index proposed in advance. At the same time, we use the same matrices $\{\mathbf{R}_i\}_{i\in[\bar{\ell}]}$ in the computation of the challenge ciphertext.

Next, in Hybrid 2, we first choose Q random subsets $\Delta_1^*, \cdots, \Delta_Q^*$ with size t according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, t)$. Then, we set matrices $\mathbf{P}_i$ for $i \in [\bar{\ell}]$ in the following way: Sample $\mathbf{J}_i^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times\ell},\rho}$ for $i \in [Q]$, and sample random matrices $\mathbf{P}_j$ from $\mathbb{Z}_q^{n\times\ell}$ for $j \in [N]$ under the constraint $\sum_{j\in\Delta_i^*} \mathbf{P}_j = \mathbf{A} \cdot \mathbf{J}_i^*$, and we denote $\sum_{j\in\Delta_i^*} \mathbf{P}_j$ as $\mathbf{P}_{\Delta_i^*}$.

Next in Hybrid 3, we set $\mathbf{P}_i$ in the following way.

1. For every $i \in [Q]$, choose a unique index δ_i that only appears in $\Delta_i^* \subset [N]$ but not the other subsets.
2. Sample $\mathbf{J}_i \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times\ell},s}$ for $i \in [N]\backslash\{\delta_1, \cdots, \delta_Q\}$. Set $\mathbf{J}_{\delta_i} = \mathbf{J}_i^* - \sum_{j\in\Delta_i^*\backslash\{\delta_i\}} \mathbf{J}_j$ for $i \in [Q]$.

3. Set $\mathbf{P}_i = \mathbf{A} \cdot \mathbf{J}_i$ for $i \in [N]$.

Moreover, in Hybrid 4, we replace the matrix $\mathbf{A}$ from the algorithm TrapGen with a randomly chosen one without trapdoor. Furthermore, in Hybrid 4, we use SampleRight to answer 0-key queries, and use Sampler-2 to answer 1-key queries. It is easy to argue the indistinguishability of the above hybrids, which is omitted here for simplicity.

Finally, we build a reduction from the N-ALS scheme to Hybrid 4, where N-ALS can be viewed as N copies of the ALS scheme with the common matrix $\mathbf{A}$. Its security has been proven in [47].

Particularly, upon receiving public keys $\mathbf{A}_{N\text{-ALS}}$ and $\{\mathbf{D}_{N\text{-ALS},i}\}_{i\in[N]}$ from the AD-CPA challenger $\mathcal{C}_{N\text{-ALS}}$, the reduction simulates the view of the adversary $\mathcal{A}_{\text{Hybrid 4}}$ in Hybrid 4 in the following way:

- Setup. The reduction sets $\mathbf{A} := \mathbf{A}^\top_{N\text{-ALS}}$, and conducts the following steps.
  1. Choose Q random subsets $\Delta_1^*, \cdots, \Delta_Q^*$ with size t according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, t)$, and choose a unique index δ_i that only appears in Δ_i^* but not the other subsets.
  2. Set $\mathbf{P}_i = \mathbf{D}^\top_{N\text{-ALS},i}$ for $i \in [N]\backslash\{\delta_1, \cdots, \delta_Q\}$.
  3. Sample $\mathbf{J}_i^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times\ell},\rho}$ and set $\mathbf{P}_i = \mathbf{A}\cdot\mathbf{J}_i^* + \mathbf{D}^\top_{N\text{-ALS},i}$ for $i \in \{\delta_1, \cdots, \delta_Q\}$.
  4. Choose $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m\times m}$ for $i \in [\bar{\ell}]$, and then compute $\mathbf{A}_i = \mathbf{A}\cdot\mathbf{R}_i - \mathsf{x}_i^* \cdot \mathbf{G} \in \mathbb{Z}_q^{n\times m}$, where $\mathsf{x}^* = (\mathsf{x}_1^*, \ldots, \mathsf{x}_{\bar{\ell}}^*) \in \{0,1\}^{\bar{\ell}}$.
  5. Send $(\mathbf{A}, \{\mathbf{A}_i\}_{i\in[\bar{\ell}]}, \{\mathbf{P}_j\}_{j\in[N]})$ to $\mathcal{A}_{\text{Hybrid 4}}$.

  In this case, we have $\sum_{j\in\Delta_i^*}(\mathbf{P}_j) = \mathbf{A}\mathbf{J}_i^* + \sum_{j\in\Delta_i^*}(\mathbf{D}^\top_{N\text{-ALS},j})$ for $i \in [Q]$.
- KeyGen for 0-key queries: The reduction uses the 0-key query algorithm as in Hybrid 4.
- KeyGen for i-th 1-key query $(C_i, \tilde{\boldsymbol{y}}_i)$ for $i \in [Q]$: The reduction asks $\mathcal{C}_{N\text{-ALS}}$ for the functional decryption key of $(\boldsymbol{y}_1^{N\text{-ALS}}, \ldots, \boldsymbol{y}_N^{N\text{-ALS}})$, where $\boldsymbol{y}_j^{N\text{-ALS}} = \tilde{\boldsymbol{y}}_i$ for all $j \in \Delta_i$, otherwise $\boldsymbol{y}_j^{N\text{-ALS}} = 0^\ell$. Then, the reduction receives $\mathsf{sk}_{\tilde{\boldsymbol{y}}}^{(i)}$. Finally, the reduction computes and returns $\mathsf{sk}_{C_i,\tilde{\boldsymbol{y}}_i} = \begin{bmatrix} \mathbf{J}_i^*\tilde{\boldsymbol{y}} + \mathsf{sk}_{\tilde{\boldsymbol{y}}_i}^{(i)} - \mathbf{R}_{\mathsf{P}_i}\mathbf{K}_2\tilde{\boldsymbol{y}}_i \\ \mathbf{K}_2\tilde{\boldsymbol{y}}_i \end{bmatrix}$[17], where $\mathbf{K}_2 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{m\times\ell},s}$, $\mathbf{R}_{C_i} = \mathsf{Eval}_{\mathsf{trap}}(\mathbf{A}, \{(\mathsf{x}_i^*, \mathbf{R}_i)\}_{i\in[\bar{\ell}]}, C_i)$ with $\mathsf{x}^* = (\mathsf{x}_1^*, \ldots, \mathsf{x}_{\bar{\ell}}^*)$.
- Challenge ciphertext query for $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ from $\mathcal{A}_{\text{Hybrid 4}}$: For $(\boldsymbol{x}_0, \boldsymbol{x}_1)$, the reduction conducts the following steps:
  1. Run the same secret key sharing as real encryption algorithm to obtain $(\{\boldsymbol{x}'_{0,i}\}_{i\in[N]}, \{\boldsymbol{x}'_{1,i}\}_{i\in[N]})$.
  2. Receive $(\mathsf{ct}_1^{N\text{-ALS}}, \mathsf{ct}_{2,1}^{N\text{-ALS}}, \ldots, \mathsf{ct}_{2,N}^{N\text{-ALS}})$ from $\mathcal{C}_{N\text{-ALS}}$ as the challenge ciphertext of N-ALS.
  3. Set $\mathsf{ct}_1 = [\mathbf{I}_m|\mathbf{R}_1|\cdots|\mathbf{R}_{\bar{\ell}}]^\top \cdot \mathsf{ct}_1^{N\text{-ALS}}$.
  4. Choose random vector $e'_{2,i} \in \mathcal{D}_{\mathbb{Z}^\ell,2\sigma\tau}$, and compute $\mathsf{ct}_{2,i} = \mathsf{ct}_{2,i}^{N\text{-ALS}} + \boldsymbol{e}'_{2,i}$ for $i \in [N]\backslash\{\delta_1, \cdots, \delta_Q\}$.

[17] Notice that the distributions of this secret keys are still statistically close to that of the direct usage of Sampler-2, since $\mathbf{J}_i^*\tilde{\boldsymbol{y}} + \mathsf{sk}_{\tilde{\boldsymbol{y}}_i}^{(i)}$ and $\mathbf{J}_i^*\tilde{\boldsymbol{y}}$ are statistically close, due to noise smudging for our parameter setting.

5. Run the algorithm ReRand in Lemma 2.10, i.e., $\mathsf{ReRand}(\mathbf{J}_i^*, \mathsf{ct}_1^{N\text{-}\mathsf{ALS}}, \sigma, \tau)$, to obtain $\mathsf{ct}_{1,i} \in \mathbb{Z}_q^m$ for $i \in [Q]$.
6. Compute $\mathsf{ct}_{2,\delta_i} = \mathsf{ct}_{1,i} + \mathsf{ct}_{2,\delta_i}^{N\text{-}\mathsf{ALS}}$ for $i \in [Q]$.
7. Send $(\mathsf{ct}_1, \mathsf{ct}_{2,1}, \cdots, \mathsf{ct}_{2,N})$ to $\mathcal{A}_{\mathsf{Hybrid}\ 4}$.

Next, let us analyze the above reduction process. Clearly, according to Lemma 2.10, for $\mathsf{ct}_1^{N\text{-}\mathsf{ALS}} = \mathbf{A}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_1 \in \mathbb{Z}_q^m$, we have $\mathsf{ct}_{1,i} = (\mathbf{A}\mathbf{J}_i^*)^\top \cdot \boldsymbol{s} + \boldsymbol{e}'_{1,i} \in \mathbb{Z}_q^\ell$, where $\boldsymbol{e}'_{1,i} \in \mathcal{D}_{\mathbb{Z}^\ell, 2\sigma\tau}$. Hence, it holds

$$\mathsf{ct}_{2,\delta_i} = \mathsf{ct}_{1,i} + \mathsf{ct}_{2,\delta_i}^{N\text{-}\mathsf{ALS}} = [\mathbf{A}\mathbf{J}_i^* + \mathbf{P}_{\delta_i}]^\top \cdot \boldsymbol{s} + \boldsymbol{e}'_{1,i} + \boldsymbol{e}'_{2,\delta_i},$$

where $\boldsymbol{e}'_{2,\delta_i}$ denotes the error vector in $\mathsf{ct}_{2,\delta_i}^{N\text{-}\mathsf{ALS}}$, which is chosen from $\mathcal{D}_{\mathbb{Z}^\ell,\sigma}$. Thus, the error vector $(\boldsymbol{e}'_{1,i} + \boldsymbol{e}'_{2,\delta_i})$ in ct_{2,δ_i} is statistically close to $\mathcal{D}_{\mathbb{Z}^\ell, 2\sigma\tau+\sigma}$. Similarly, for $i \in [N] \backslash \{\delta_1, \cdots, \delta_Q\}$, the error vector in $\mathsf{ct}_{2,i}$ is statistically close to $\mathcal{D}_{\mathbb{Z}^\ell, 2\sigma\tau+\sigma}$ as well.

Hence, the reduction faithfully simulates Hybrid 4. Finally, it is not hard to verify that the reduction translates the key queries of adversary in Hybrid 4 into an admissible set of queries to N-ALS. Thus, the reduction would break N-ALS with a non-negligible probability if the adversary has a non-negligible probability in Hybrid 4. □

By combining Theorem 7.7 with the technique in Section 5.3, we can obtain the following corollary.

Corollary 7.8 (SA-IND Security) *Assuming the hardness of* LWE *for appropriate* q, α, n*, then there exists an* AB-FEIP *that is* SA-IND *secure.*

Acknowledgements. We would like to thank the anonymous reviewers of Eurocrypt 2021 for their insightful advices. Qiqi Lai is supported by the National Natural Science Foundation of China (62172266, 61802241, U2001205), the National Cryptography Development Foundation during the 13th Five-year Plan Period (MMJJ20180217), and the Fundamental Research Funds for the Central Universities (GK202103093). Feng-Hao Liu and Zhedong Wang are supported by the NSF Career Award CNS-1942400. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

References

1. M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. To be published in Asiacrypt, 2020. `https://eprint.iacr.org/2020/577`.
2. S. Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 3–35. Springer, Heidelberg, Aug. 2017.
3. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [31], pages 553–572.

4. S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 500–518. Springer, Heidelberg, Aug. 2013.
5. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Robshaw and Katz [46], pages 333–362.
6. S. Agrawal and A. Rosen. Functional encryption for bounded collusions, revisited. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 173–205. Springer, Heidelberg, Nov. 2017.
7. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
8. J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In Garay and Gennaro [25], pages 297–314.
9. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2010.
10. P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. From selective to adaptive security in functional encryption. In Gennaro and Robshaw [28], pages 657–677.
11. P. Ananth and V. Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Heidelberg, Dec. 2019.
12. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.
13. D. Boneh, T. Roughgarden, and J. Feigenbaum, editors. *45th ACM STOC*. ACM Press, June 2013.
14. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, Mar. 2011.
15. X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, Heidelberg, May 2010.
16. Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. `https://eprint.iacr.org/2020/1024`.
17. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, Jan. 2012.
18. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In Boneh et al. [13], pages 575–584.
19. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, Oct. 2011.
20. Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In M. Naor, editor, *ITCS 2014*, pages 1–12. ACM, Jan. 2014.
21. Z. Brakerski and V. Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Robshaw and Katz [46], pages 363–384.

22. R. Canetti and Y. Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 446–476. Springer, Heidelberg, Apr. / May 2017.
23. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [31], pages 523–552.
24. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In Garay and Gennaro [25], pages 335–352.
25. J. A. Garay and R. Gennaro, editors. *CRYPTO 2014, Part I*, volume 8616 of *LNCS*. Springer, Heidelberg, Aug. 2014.
26. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, Oct. 2013.
27. R. Gay and R. Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020. `https://eprint.iacr.org/2020/1010`.
28. R. Gennaro and M. J. B. Robshaw, editors. *CRYPTO 2015, Part II*, volume 9216 of *LNCS*. Springer, Heidelberg, Aug. 2015.
29. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
30. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, Aug. 2013.
31. H. Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May / June 2010.
32. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In Boneh et al. [13], pages 555–564.
33. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, Aug. 2012.
34. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In Boneh et al. [13], pages 545–554.
35. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In Gennaro and Robshaw [28], pages 503–523.
36. S. Gorbunov and D. Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Heidelberg, Nov. / Dec. 2015.
37. R. Goyal, V. Koppula, and B. Waters. Semi-adaptive security and bundling functionalities made generic and easy. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388. Springer, Heidelberg, Oct. / Nov. 2016.
38. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. `https://eprint.iacr.org/2020/1003`.
39. S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In J. H. Cheon and

T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Heidelberg, Dec. 2016.
40. Q. Lai, F.-H. Liu, and Z. Wang. Almost tight security in lattices with polynomial moduli - PRF, IBE, all-but-many LTF, and more. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 652–681. Springer, Heidelberg, May 2020.
41. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, Apr. 2012.
42. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, Oct. 2004.
43. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint archive, Report 2010/556, 2010. `https://eprint.iacr.org/2010/556`.
44. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.
45. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.
46. M. Robshaw and J. Katz, editors. *CRYPTO 2016, Part III*, volume 9816 of *LNCS*. Springer, Heidelberg, Aug. 2016.
47. Z. Wang, X. Fan, and F.-H. Liu. FE for inner products and its application to decentralized ABE. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 97–127. Springer, Heidelberg, Apr. 2019.

A Supplementary Material of Section 5.1

In this section, we first argue the correctness of $(1, \mathsf{poly})$-PHPE in Section A.1. Then, we set the parameters so that correctness and security of the scheme are satisfied in Section A.2. Finally, we present the security proof of $(1, \mathsf{poly})$-PHPE in Section A.3.

A.1 Correctness of $(1, \mathsf{poly})$-PHPE in Section 5.1

The correctness of the scheme follows from our choice of parameters. Specifically, to show correctness we first note that

$$\begin{aligned}\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} &= \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}, \{\mathbf{B}_j, \boldsymbol{v}_j\}, \widehat{C}\circ\mathsf{IP}, \boldsymbol{y}\big)\\ &= \big(\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \widehat{C}\circ\mathsf{IP}(\boldsymbol{x},\boldsymbol{y})\cdot\mathbf{G}\big)^\top \cdot \boldsymbol{s} + \boldsymbol{e}_{\mathsf{Eval}}.\end{aligned}$$

When $\widehat{C}\circ\mathsf{IP}(\boldsymbol{x},\boldsymbol{y}) = \gamma$, we have $\begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}\end{pmatrix} = \begin{bmatrix}\mathbf{A}^\top\\ (\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\cdot\mathbf{G})^\top\end{bmatrix}\cdot\boldsymbol{s} + \begin{pmatrix}\boldsymbol{e}\\ \boldsymbol{e}_{\mathsf{Eval}}\end{pmatrix}$. Hence, it holds $\mathbf{K}^\top\cdot\begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}\end{pmatrix} = \mathbf{P}^\top\cdot\boldsymbol{s} + \mathbf{K}^\top\cdot\begin{pmatrix}\boldsymbol{e}\\ \boldsymbol{e}_{\mathsf{Eval}}\end{pmatrix}$. Therefore, we know $\boldsymbol{\beta}_1 - \mathbf{K}^\top\cdot\begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}\end{pmatrix} = \boldsymbol{b} + \boldsymbol{e}' - \mathbf{K}^\top\cdot\begin{pmatrix}\boldsymbol{e}\\ \boldsymbol{e}_{\mathsf{Eval}}\end{pmatrix}$.

Thus, when $\widehat{C}\circ\mathsf{IP}(\boldsymbol{x},\boldsymbol{y}) = \gamma$, we require the first $m-1$ coordinates of $\boldsymbol{e}' - \mathbf{K}^\top\cdot\begin{pmatrix}\boldsymbol{e}\\ \boldsymbol{e}_{\mathsf{Eval}}\end{pmatrix}$ to be bounded by $q/4$, which can be ensured by our parameter setting below.

Otherwise, if $\widehat{C}\circ\mathsf{IP}(\boldsymbol{x},\boldsymbol{y}) = \gamma' \neq \gamma \bmod q$, then setting $\gamma' = \gamma + \gamma^*$ for some γ^*, and parsing $\mathbf{K} = \begin{bmatrix}\mathbf{J}+\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix}$, we have that

$$\boldsymbol{\eta} = \boldsymbol{\beta}_1 - \mathbf{K}^\top\cdot\begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}\end{pmatrix} = \boldsymbol{b} + \gamma^*\cdot\mathbf{K}_2^\top\mathbf{G} + \boldsymbol{e}^*,$$

for some error $\boldsymbol{e}^*$. Hence, with negligible probability all first $m-1$ coefficients of $\boldsymbol{\eta}$ will be smaller than $q/4$.

A.2 Parameter Setting of $(1, \mathsf{poly})$-PHPE in Section 5.1

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 $\mathsf{TrapGen}$ in Theorem 2.13.
3. We must choose s_B such that LWE_{q,n,s_B} assumption holds.
4. We set s used in $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ such that the output matrices are statistically indistinguishable.
5. We set s and ρ to meet the requirements in Theorem 4.2.

6. We set s_D such that $\boldsymbol{e}' \overset{s}{\approx} \mathbf{J}^\top \cdot \boldsymbol{e} + \boldsymbol{e}'$.

To satisfy the constraints, we can choose $n = \mathsf{poly}(\lambda)$, $m = O(n \log q)$, $s_B = \sqrt{n}$, $s = O(\ell n \log q)^{O(d)} \cdot \omega(\sqrt{\log m})$, $\rho = s\sqrt{m} \cdot O(\ell n \log q)^{O(d)} \cdot \lambda^{\omega(1)}$, $s_D = \rho\sqrt{mn} \cdot \lambda^{\omega(1)}$, $q = \rho m \sqrt{n} \cdot \lambda^{\omega(1)}$. Note that there are other possible choices of the parameters. Here we just set one for demonstration, showing that parameters are feasible under the constraints.

A.3 Security of (1, poly)-PHPE in Section 5.1

Theorem (Restatement of Theorem 5.3) *Assuming the hardness of* LWE*, then the scheme described in Section 5.1 is a* PHPE *for the class $\mathcal{F}$, achieving* $(1, \mathsf{poly})$-Sel-Sim *security that allows at most one* 1*-key pre-challenge query (and an unbounded polynomial number of* 0*-keys for both pre and post-challenge queries), according to Definition 2.17.*

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 $\mathsf{Sim}(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|}, b, \mathsf{List})$:

1. $\mathsf{Sim}_1(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all public parameters as in the real PH.Setup
2. $\mathsf{Sim}_2(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all keys using the real PH.KeyGen
3. $\mathsf{Sim}_3(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|}, b, \mathsf{List})$: It takes as input the public attributes $\boldsymbol{y}$, the size of the private attributes $\boldsymbol{x}$, a message b, and a list List. It constructs the challenge ciphertext as follows.
   - It samples $\boldsymbol{\beta}_0, \boldsymbol{u}_i, \boldsymbol{v}_j$ independently and uniformly from $\mathbb{Z}_q^m$. If $(b, \mathsf{List}) = \perp$, it samples $\boldsymbol{\beta}_1$ randomly from $\mathbb{Z}_q^m$
   - If $(b, \mathsf{List}) = (\mu, \widehat{C^*} \circ \mathsf{IP}_\gamma)$, this means that the adversary has already queried the 1-key $\widehat{C^*} \circ \mathsf{IP}_\gamma$ and Sim_2 has already generated the secret key $\mathsf{sk}_{\widehat{C^*} \circ \mathsf{IP}_\gamma} := \mathbf{K}^*$. Now Sim_3 generates $\boldsymbol{\beta}_1$ to satisfy the decryption consistency as follows.
     - Let $\boldsymbol{u}_{\widehat{C^*} \circ \mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i \in [\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j \in [t]}, \widehat{C^*} \circ \mathsf{IP}, \boldsymbol{y}\big)$.
     - Set $\boldsymbol{\beta}_1 = {\mathbf{K}^*}^\top \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C^*} \circ \mathsf{IP}} \end{pmatrix} + \boldsymbol{e}' + \boldsymbol{b}$, for $\boldsymbol{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$ and $\boldsymbol{b} = [0 \cdots 0, \lceil q/2 \rceil \mu]^\top \in \mathbb{Z}_q^m$.
   - It outputs the simulated ciphertext

$$\mathsf{ct}^* = \big(\{\boldsymbol{u}_i\}_{i \in [\ell]}, \{\boldsymbol{v}_j\}_{j \in [t]}, \boldsymbol{y}, \boldsymbol{\beta}_0, \boldsymbol{\beta}_1\big)$$

4. $\mathsf{Sim}_4(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all keys using the real PH.KeyGen.

Auxiliary Algorithms.

$\mathsf{PH.Setup}_1^*(1^\lambda, 1^d, \boldsymbol{x}, \boldsymbol{y})$: Do the following:

1. Sample a random matrix with associated trapdoor:

$$(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$$

2. Let $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - y_i \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $i \in [\ell]$, where $\mathbf{R}_i \leftarrow \{-1, 1\}^{m \times m}$ for $i \in [\ell]$.
3. Let $\mathbf{B}_j = \mathbf{A} \cdot \mathbf{R}_j^{'} - x_j \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $j \in [t]$, where $\mathbf{R}_j^{'} \leftarrow \{-1, 1\}^{m \times m}$ for $j \in [t]$.
4. Sample $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m}, \sqrt{\rho^2 + s^2}}$ and set $\mathbf{P} = \mathbf{A} \cdot \mathbf{J} (\mathrm{mod} q)$.
5. Output the master public key as $\mathsf{PH.mpk} = \big(\{\mathbf{A}_i\}_{i \in [\ell]}, \{\mathbf{B}_j\}_{j \in [t]}, \mathbf{A}, \mathbf{P}\big)$ and the master secret key as $\mathsf{PH.msk} = \big(\mathbf{T}_\mathbf{A}, \{\mathbf{R}_i\}_{i \in [\ell]}, \{\mathbf{R}_j^{'}\}_{j \in [t]}, \mathbf{J}\big)$.

$\mathsf{PH.Enc}_1^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: The ciphertext is computed as follows:

1. Sample $\boldsymbol{s}, \boldsymbol{e} \leftarrow D_{\mathbb{Z}^n, s_B}$ and set $\boldsymbol{\beta}_0 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}$.
2. For $i \in [\ell], j \in [t]$ compute $\boldsymbol{u}_i = \mathbf{R}_i^\top \cdot \boldsymbol{\beta}_0, \quad \boldsymbol{v}_j = (\mathbf{R}_j^{'})^\top \cdot \boldsymbol{\beta}_0$, where the matrices $\{\mathbf{R}_i^\top\}$ and $\{\mathbf{R}_j^{'}\}$ are the matrices in the msk generated by $\mathsf{PH.Setup}_1^*$.
3. $\boldsymbol{\beta}_1$ is computed as real encryption algorithm.
4. Output $\big(\{\boldsymbol{u}_i\}_{i \in [\ell]}, \{\boldsymbol{v}_j\}_{j \in [t]}, \boldsymbol{y}, \boldsymbol{\beta}_0, \boldsymbol{\beta}_1\big)$.

$\mathsf{PH.KeyGen}_1^*(\mathsf{PH.msk}, \widehat{C} \circ \mathsf{IP}_\gamma)$: Do the following:

1. Compute the homomorphic public key corresponding to circuit $\widehat{C} \circ \mathsf{IP}$ as

$$\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} = \mathsf{Eval}_{\mathsf{pk}}\big(\{\mathbf{A}_i\}_{i \in [\ell]}, \{\mathbf{B}_j\}_{j \in [t]}, \widehat{C} \circ \mathsf{IP}\big)$$

2. We know that $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} = \mathbf{A} \cdot \mathbf{R}_{\widehat{C} \circ \mathsf{IP}} - \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y}) \rangle \cdot \mathbf{G}$. For different types of the key query, we respond as follows:
   - 0-key query $\widehat{C} \circ \mathsf{IP}_\gamma$ such that $\widehat{C} \circ \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) \neq \gamma$. In this case, we have:

$$\big[\mathbf{A} | \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}\big] = [\mathbf{A} | \mathbf{A} \cdot \mathbf{R}_{\widehat{C} \circ \mathsf{IP}} + (\gamma - \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y}) \rangle) \cdot \mathbf{G}\big].$$

     As the $\mathbf{G}$ trapdoor does not vanish, the algorithm runs

$$\mathbf{K} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \mathbf{R}_{\widehat{C}_i \circ \mathsf{IP}}, \gamma - \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y}) \rangle, \mathbf{P}, s),$$

     satisfying $\big[\mathbf{A} | \mathbf{A} \cdot \mathbf{R}_{\widehat{C}_i \circ \mathsf{IP}} + (\gamma - \langle \mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y}) \rangle) \cdot \mathbf{G}\big] \cdot \mathbf{K} = \mathbf{P}$.
   - 1-key query $\widehat{C} \circ \mathsf{IP}_\gamma$ such that $\widehat{C} \circ \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) = \gamma$. In this case, the algorithm computes

$$\mathbf{R}_{\widehat{C} \circ \mathsf{IP}} = \mathsf{Eval}_{\mathsf{trap}}\big(\{\mathbf{R}_i\}_{i \in [\ell]}, \{\mathbf{R}_j^{'}\}_{j \in [t]}, \mathbf{A}, \boldsymbol{y}, \widehat{C} \circ \mathsf{IP}\big),$$

     samples $\mathbf{K}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m}, s}$, and sets $\mathbf{K} = \begin{bmatrix} \mathbf{J} - \mathbf{R}_{\widehat{C} \circ \mathsf{IP}} \cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix}$. Note that by construction, we have

$$\big[\mathbf{A} | \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}\big] \cdot \mathbf{K} = \big[\mathbf{A} | \mathbf{A} \cdot \mathbf{R}_{\widehat{C} \circ \mathsf{IP}}\big] \cdot \mathbf{K} = \mathbf{P}.$$

3. Return $\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma} = \mathbf{K}$.

$\mathsf{PH.Enc}_2^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: The ciphertext is computed as follows:

1. Generate $\boldsymbol{\beta}_0, \{\boldsymbol{u}_i\}, \{\boldsymbol{v}_i\}$ as $\mathsf{PH.Enc}_1^*$.
2. Sample $\boldsymbol{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_D}$, set $\boldsymbol{b} = (0, \cdots, 0, \lceil q/2 \rceil \mu)^\top \in \mathbb{Z}_q^m$ and compute $\boldsymbol{\beta}_1$ as follows:
   - If there is no 1-query before the challenge ciphertext, it computes $\boldsymbol{\beta}_1 = \mathbf{J}^\top \cdot \boldsymbol{\beta}_0 + \boldsymbol{e}' + \boldsymbol{b}$.
   - If the adversary has already queried the 1-key $\widehat{C^*} \circ \mathsf{IP}_\gamma$, it computes $\boldsymbol{\beta}_1 = (\mathsf{sk}_{\widehat{C^*}\circ\mathsf{IP}_\gamma})^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C^*}\circ\mathsf{IP}} \end{pmatrix} + \boldsymbol{e}' + \boldsymbol{b}$, where $\boldsymbol{u}_{\widehat{C^*}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C^*} \circ \mathsf{IP}, \boldsymbol{y}\big)$.
3. Output $\big(\{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}, \boldsymbol{y}, \boldsymbol{\beta}_0, \boldsymbol{\beta}_1\big)$.

$\mathsf{PH.Enc}_3^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: Sample $\boldsymbol{\beta}_0 \leftarrow \mathbb{Z}_q^m$ randomly. Compute the remaining ciphertext elements exactly the same as in $\mathsf{PH.Enc}_2^*$.

$\mathsf{PH.KeyGen}_2^*(\mathsf{PH.msk}, \widehat{C} \circ \mathsf{IP}_\gamma)$: Do the following:

1. If $\widehat{C} \circ \mathsf{IP}_\gamma$ is a 0-key query, sample the key $\mathbf{K}$ using the $\mathsf{SampleLeft}$ algorithm as in KeyGen.
2. If $\widehat{C} \circ \mathsf{IP}_\gamma$ is a 1-key query, set $\mathbf{K}$ the same as $\mathsf{PH.KeyGen}_1^*$.
3. Return $\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma} = \mathbf{K}$.

$\mathsf{PH.Enc}_4^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: If there is no 1-query before the challenge ciphertext, it samples $\boldsymbol{\beta}_1$ randomly from $\mathbb{Z}_q^m$. Otherwise, Compute ciphertext elements the same as $\mathsf{PH.Enc}_3^*$.

$\mathsf{PH.Enc}_5^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \widehat{C^*} \circ \mathsf{IP})$: Sample $\boldsymbol{u}_i, \boldsymbol{v}_i$ randomly. Everything else remains the same as $\mathsf{PH.Enc}_4^*$.

Now we establish the proof via the following hybrids, where the first one corresponds to the real world and the last corresponds to the ideal world.

Hybrids.

Hybrid 0: The real experiment.

Hybrid 1: The real game algorithms $\mathsf{PH.Setup}$ and $\mathsf{PH.Enc}$ are replaced with $\mathsf{PH.Setup}_1^*$ and $\mathsf{PH.Enc}_1^*$, which use the knowledge of $(\boldsymbol{x}, \boldsymbol{y})$ to generate the public parameters, the master public/secret keys, and additionally sets $\mathbf{P} = \mathbf{A} \cdot \mathbf{J}$.

Hybrid 2: The real game $\mathsf{PH.KeyGen}$ is replaced with $\mathsf{PH.KeyGen}_1^*$ where instead of using the trapdoor $\mathbf{T_A}$ of the matrix $\mathbf{A}$, secret keys for a 0-key queries are sampled using the public trapdoor $\mathbf{T_G}$ along with the trapdoor information generated in $\mathsf{PH.Setup}_1^*$, and the secret key for the 1-key query for function $\widehat{C^*} \circ \mathsf{IP}_\gamma$ is computed as $\mathbf{K} = \begin{bmatrix} \mathbf{J} - \mathbf{R}_{\widehat{C^*}\circ\mathsf{IP}} \cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{bmatrix}$.

Hybrid 3: $\mathsf{PH.Enc}_1^*$ is replaced by $\mathsf{PH.Enc}_2^*$, in which $\boldsymbol{\beta}_1$ is computed from $\mathbf{J}$ if there is no 1-key queried before, or otherwise from the given 1-key.

Hybrid 4: The encryption algorithm is changed from $\mathsf{PH.Enc}_2^*$ to $\mathsf{PH.Enc}_3^*$. Here, the ciphertext element $\boldsymbol{\beta}_0$ is switched to random and all other ciphertext elements are derived from it.

Hybrid 5: The algorithm $\mathsf{PH.KeyGen}_1^*$ is replaced by $\mathsf{PH.KeyGen}_2^*$. The $\mathsf{PH.KeyGen}_2^*$ algorithm is as the real $\mathsf{PH.KeyGen}$ algorithm, except for the response to the 1-key query $\widehat{C}^* \circ \mathsf{IP}_\gamma$.

Hybrid 6: The encryption algorithm is changed from $\mathsf{PH.Enc}_3^*$ to $\mathsf{PH.Enc}_4^*$. Here, the ciphertext element $\boldsymbol{\beta}_1$ is random or computed by using $\mathsf{sk}_{\widehat{C}^* \circ \mathsf{IP}_\gamma}$, depending on whether a 1-key has been queried before the challenge ciphertext.

Hybrid 7: The key generation algorithm is changed from $\mathsf{PH.KeyGen}_2^*$ to the real $\mathsf{PH.KeyGen}$ algorithm.

Hybrid 8: The encryption algorithm is changed from $\mathsf{PH.Enc}_4^*$ to $\mathsf{PH.Enc}_5^*$, in which the ciphertext elements $\{\boldsymbol{u}_i\}, \{\boldsymbol{v}_i\}$ are changed to random. The remaining elements $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$ are as before.

Hybrid 9: The algorithm $\mathsf{PH.Setup}_1^*$ is replaced by the real $\mathsf{PH.Setup}$ algorithm. This hybrid is identical to the ideal experiment when running the simulator Sim.

Below we show that each two adjacent hybrids are indistinguishable.

Lemma A.1 Hybrid 0 *and* Hybrid 1 *are statistically indistinguishable.*

Proof. The difference between the two hybrids is in how the public parameters and the ciphertext are generated. In Hybrid 0, for $\mathsf{PH.mpk}$,

$$\{\mathbf{A}_i\}, \{\mathbf{B}_j\}, \mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{n \times m} \quad \forall i \in [\ell], j \in [t],$$

and for ct^*,

$$\boldsymbol{\beta}_0 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}, \boldsymbol{u}_i = (\mathbf{A}_i + y_i \cdot \mathbf{G})^\top \boldsymbol{s} + \mathbf{R}_i^\top \boldsymbol{e}, \boldsymbol{v}_i = (\mathbf{B}_i + x_i \cdot \mathbf{G})^\top \boldsymbol{s} + (\mathbf{R}_i')^\top \boldsymbol{e},$$

In Hybrid 1, for $\mathsf{PH.mpk}$,

$$\{\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - y_i \cdot \mathbf{G}\}, \{\mathbf{B}_j = \mathbf{A} \cdot \mathbf{R}_j^{'} - x_j \cdot \mathbf{G}\}, \mathbf{P} = \mathbf{A} \cdot \mathbf{J} \quad \forall i \in [\ell], j \in [t],$$

and for ct^*,

$$\boldsymbol{\beta}_0 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}, \boldsymbol{u}_i = \mathbf{R}_i^\top \cdot \boldsymbol{\beta}_0, \boldsymbol{v}_i = (\mathbf{R}_i')^\top \boldsymbol{\beta}_0,$$

We argue that the joint distribution of the public parameters and ciphertext

$$(\mathbf{A}, \{\mathbf{A}_i, \boldsymbol{u}_i\}_{i \in [\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j \in [t]}, \mathbf{P})$$

is statistically indistinguishable between the two hybrids.

On the one hand, by Theorem 2.13, we have that $\mathbf{A} \overset{\$}{\approx} \mathcal{U}$ for $\mathbf{A}$ output by $\mathsf{TrapGen}$ algorithm. Then by Lemma 2.16, we have that,

$$(\mathbf{A}, \mathbf{A} \cdot \mathbf{R}_i - y_i \cdot \mathbf{G}, \mathbf{R}_i^\top \boldsymbol{e}) \overset{\$}{\approx} (\mathbf{A}, \mathcal{U}, \mathbf{R}_i^\top \boldsymbol{e}),$$

and this holds for $\mathbf{B}_j = \mathbf{A} \cdot \mathbf{R}_j' - x_j \cdot \mathbf{G}$ as well. And since for all i, $\mathbf{R}_i$ (resp. $\mathbf{R}_i'$) is randomly and independently chosen, it follows that

$$(\mathbf{A}, \{\mathbf{A} \cdot \mathbf{R}_i - y_i \cdot \mathbf{G}\}, \{\mathbf{R}_i^\top \boldsymbol{e}\}, \{\mathbf{A} \cdot \mathbf{R}_j' - x_j \cdot \mathbf{G}\}, \{(\mathbf{R}_j')^\top \boldsymbol{e}\}) \overset{\$}{\approx} (\mathbf{A}, \{\mathcal{U}_i\}, \{\mathbf{R}_i^\top \boldsymbol{e}\}, \{\mathcal{U}_j\}, \{(\mathbf{R}_j')^\top \boldsymbol{e}\}).$$

On the other hand, by Lemma 2.7, we have $(\mathbf{A}, \mathbf{A}\cdot\mathbf{J}) \stackrel{\$}{\approx} (\mathbf{A}, \mathcal{U})$. Furthermore, the ciphertext components $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are derived simply by adding $(\mathbf{A}_i + y_i\cdot\mathbf{G})^\top \cdot \boldsymbol{s}$ and $(\mathbf{B}_j + x_j \cdot \mathbf{G})^\top \cdot \boldsymbol{s}$ to $\mathbf{R}_i^\top \boldsymbol{e}$ and $(\mathbf{R}'_j)^\top \boldsymbol{e}$, respectively. Since applying a function to two statistically indistinguishable distributions produces two statistically indistinguishable distributions, this shows that the public parameters, the ciphertext and the secret keys are statistically close in both hybrids. □

Lemma A.2 Hybrid 1 *and* Hybrid 2 *are statistically indistinguishable.*

Proof. Between Hybrid 1 and Hybrid 2, the manner of generating keys changes from $\mathsf{PH.KeyGen}$ to $\mathsf{PH.KeyGen}_1^*$. We consider two cases:

1. For the 0-key $\widehat{C}\circ\mathsf{IP}$, in Hybrid 1, these keys are sampled using the SampleLeft algorithm, whereas in Hybrid 2, they are sampled using the SampleRight algorithm. By Lemma 2.15 and our setting of parameters, the resulting distributions are statistically indistinguishable.
2. For the 1-key $\widehat{C}\circ\mathsf{IP}_\gamma$, we note that $\mathbf{P}$ is chosen the same way in both Hybrid 1 and Hybrid 2. In Hybrid 1, we sample $\mathbf{K} = \begin{bmatrix}\mathbf{J}' + \mathbf{K}_1 \\ \mathbf{K}_2\end{bmatrix}$ by first sampling $\mathbf{J}' \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$ then using SampleLeft such that

$$\left[\mathbf{A}|\mathbf{A}_{\widehat{C^*}\circ\mathsf{IP}} + \gamma\cdot\mathbf{G}\right]\cdot\begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{P} - \mathbf{A}\mathbf{J}',$$

where the marginal distribution of $\mathbf{P} = \mathbf{A}\cdot\mathbf{J}'$ is uniformly at random, as $\mathbf{J}'$ is hidden in the view of adversary in this case.
In Hybrid 2, we output $\mathbf{K} = \begin{bmatrix}\mathbf{J} - \mathbf{R}_{\widehat{C}\circ\mathsf{IP}}\cdot\mathbf{K}_2\\ \mathbf{K}_2\end{bmatrix}$, where $\mathbf{K}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$, $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$, and

$$\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{trap}}\big(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_i^{'}\}_{i\in[t]}, \mathbf{A}, \boldsymbol{y}, \widehat{C}\circ\mathsf{IP}, \boldsymbol{y}\big).$$

To rephrase in the terms as we used in Section 4, the key generation (with the public $\mathbf{A}, \mathbf{P}$) in Hybrid 1 is exactly the procedure of Sampler-1, and in Hybrid 2 the Sampler-2. Therefore, by Theorem 4.2, we know that the two hybrids are statistically indistinguishable.

This completes the proof. □

Lemma A.3 Hybrid 2 *and* Hybrid 3 *are statistically indistinguishable.*

Proof. The only difference between the two hybrids is the way how the ciphertext element $\boldsymbol{\beta}_1$ is generated. In Hybrid 2, we have $\boldsymbol{\beta}_1 = \mathbf{P}^\top\boldsymbol{s} + \boldsymbol{e}' + \boldsymbol{b}$. In Hybrid 3, for $\boldsymbol{\beta}_1$, if there is no 1-key query before challenge, we have

$$\begin{aligned}\boldsymbol{\beta}_1 =& \mathbf{J}^\top\cdot\boldsymbol{\beta}_0 + \boldsymbol{e}' + \boldsymbol{b}\\ =& (\mathbf{A}\mathbf{J})^\top\boldsymbol{s} + \mathbf{J}^\top\cdot\boldsymbol{e} + \boldsymbol{e}' + \boldsymbol{b}\\ =& \mathbf{P}^\top\boldsymbol{s} + \mathbf{J}^\top\cdot\boldsymbol{e} + \boldsymbol{e}' + \boldsymbol{b}.\end{aligned}$$

Thus, it suffices to show that $\boldsymbol{e}' \overset{s}{\approx} \mathbf{J}^\top \cdot \boldsymbol{e} + \boldsymbol{e}'$. By the smudging lemma, i.e., Lemma 2.9 and our setting of parameters, we achieve the goal. If the adversary has already queried the 1-key $\widehat{C^*} \circ \mathsf{IP}_\gamma$, we have $\boldsymbol{u}_{\widehat{C^*} \circ \mathsf{IP}} = \mathbf{R}^\top_{\widehat{C^*} \circ \mathsf{IP}} \cdot \boldsymbol{\beta}_0$, and thus,

$$\begin{aligned}\boldsymbol{\beta}_1 =& \begin{pmatrix} \mathbf{J} - \mathbf{R}_{\widehat{C} \circ \mathsf{IP}} \cdot \mathbf{K}_2 \\ \mathbf{K}_2 \end{pmatrix}^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C^*} \circ \mathsf{IP}} \end{pmatrix} + \boldsymbol{e}' + \boldsymbol{b} \\ =& \mathbf{J}^\top \cdot \boldsymbol{\beta}_0 + \boldsymbol{e}' + \boldsymbol{b},\end{aligned}$$

Thus we complete the proof. □

Lemma A.4 Hybrid 3 *and* Hybrid 4 *are computationally indistinguishable assuming* LWE *is hard.*

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 $(\mathbf{A}, \boldsymbol{z})$ where $\boldsymbol{z}$ is either real or random. The reduction does as follows:

1. Run $\mathsf{PH.Setup}^*_1$ and $\mathsf{PH.KeyGen}^*_1$. These algorithms are used to produce the public parameters and the function keys. We note that these two algorithms can be implemented without the trapdoor of $\mathbf{A}$.
2. To produce the ciphertext, set $\boldsymbol{\beta}_0 = \boldsymbol{z}$ and generate the remaining components as in Hybrid 3.

Now if $\boldsymbol{z} = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}$, then the reduction faithfully simulates the distribution of Hybrid 3, whereas if $\boldsymbol{z}$ is random, the reduction simulates Hybrid 4. This completes the proof. □

Lemma A.5 Hybrid 4 *and* Hybrid 5 *are statistical indistinguishable.*

Proof. The proof is analogous to the proof of indistinguishability between Hybrid 1 and Hybrid 2 for generating secret keys of 0-key queries. □

Lemma A.6 Hybrid 5 *and* Hybrid 6 *are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way of generating ciphertext element $\boldsymbol{\beta}_1$ in the case that there is no 1-key queried.

We know that in Hybrid 5, if there is no 1-key queried, $\boldsymbol{\beta}_1 = \mathbf{J}^\top \cdot \boldsymbol{\beta}_0 + \boldsymbol{e}' + \boldsymbol{b}$, and in Hybrid 6, it samples $\boldsymbol{\beta}_1$ randomly from $\mathbb{Z}_q^m$. In this case, we need to show $\boldsymbol{\beta}_1 \overset{s}{\approx} \mathcal{U}$ in Hybrid 5. By the leftover hash Lemma 2.7, $\mathbf{J}^\top \cdot \boldsymbol{u} \overset{s}{\approx} \mathcal{U}$ for a uniformly random vector $\boldsymbol{u} \in \mathbb{Z}_q^m$, thus $\boldsymbol{\beta}_1 \overset{s}{\approx} \mathcal{U}$, when $\mathbf{J}$ is not given. □

Lemma A.7 Hybrid 6 *and* Hybrid 7 *are statistically indistinguishable.*

Proof. The proof is analogous to the proof of indistinguishability between Hybrid 1 and Hybrid 2 for generating secret keys of 1-key query. □

Lemma A.8 Hybrid 7 *and* Hybrid 8 *are statistically indistinguishable.*

Proof. For the elements $\boldsymbol{u}_i, \boldsymbol{v}_i$, in Hybrid 7, they are computed as $\boldsymbol{u}_i = \mathbf{R}_i^\top \cdot \boldsymbol{u}$ and $\boldsymbol{v}_i = (\mathbf{R}'_i)^\top \cdot \boldsymbol{u}$. However, in Hybrid 8 these are randomly chosen from the encoding space. The indistinguishability of two hybrids follows from the leftover hash Lemma 2.16:

$$\Big(\mathbf{A}, \mathbf{B}, \boldsymbol{u}, \{\mathbf{A}\mathbf{R}_i, \mathbf{R}_i^\top \boldsymbol{u}\}, \{\mathbf{B}\mathbf{R}'_i, (\mathbf{R}'_i)^\top \boldsymbol{u}\}\Big) \overset{\$}{\approx} \Big(\mathbf{A}, \mathbf{B}, \boldsymbol{u}, \{\mathbf{A}\mathbf{R}_i, \boldsymbol{u}\}, \{\mathbf{B}\mathbf{R}'_i, \boldsymbol{u}\}\Big).$$

□

Lemma A.9 Hybrid 8 *and* Hybrid 9 *are statistically indistinguishable.*

Proof. The proof follows similarly as the proof of indistinguishability between Hybrid 0 and Hybrid 1. □

This completes the security proof. □

B Supplementary Material of Section 5.2

In this section, we first argue the correctness of (Q, poly)-PHPE in Section B.1. Then, we set the parameters so that correctness and security of the scheme are satisfied in Section B.2. Finally, we present the security proof of (Q, poly)-PHPE in Section B.3.

B.1 Correctness of (Q, poly)-PHPE in Section 5.2

The correctness of the scheme follows from our choice of parameters. Specifically, to show correctness we first note that

$$\begin{aligned}\boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} &= \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}, \{\mathbf{B}_j, \boldsymbol{v}_j\}, \widehat{C} \circ \mathsf{IP}, \boldsymbol{y}\big)\\ &= \big(\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \widehat{C} \circ \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) \cdot \mathbf{G}\big)^\top \cdot \boldsymbol{s} + \boldsymbol{e}_{\mathsf{Eval}}.\end{aligned}$$

When $\widehat{C} \circ \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) = \gamma$, we have $\begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} \end{pmatrix} = \begin{bmatrix} \mathbf{A}^\top \\ (\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma \cdot \mathbf{G})^\top \end{bmatrix} \cdot \boldsymbol{s} + \begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{e}_{\mathsf{Eval}} \end{pmatrix}$. Hence, it holds $\mathbf{K}^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} \end{pmatrix} = \mathbf{P}_\Delta^\top \cdot \boldsymbol{s} + \mathbf{K}^\top \cdot \begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{e}_{\mathsf{Eval}} \end{pmatrix}$. Therefore, we know $\sum_{k\in\Delta} \boldsymbol{\beta}_{1,k} - \mathbf{K}^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} \end{pmatrix} = \boldsymbol{b} + \sum_{k\in\Delta} \boldsymbol{e}'_k - \mathbf{K}^\top \cdot \begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{e}_{\mathsf{Eval}} \end{pmatrix}$.

Thus, when $\widehat{C} \circ \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) = \gamma$, we require the first $m-1$ coordinates of $\sum_{k\in\Delta} \boldsymbol{e}'_k - \mathbf{K}^\top \cdot \begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{e}_{\mathsf{Eval}} \end{pmatrix}$ are bounded by $q/4$, which can be ensured by our parameter setting below.

Otherwise, if $\widehat{C} \circ \mathsf{IP}(\boldsymbol{x}, \boldsymbol{y}) = \gamma' \neq \gamma \bmod q$, then setting $\gamma' = \gamma + \gamma^*$ for some γ^*, and parsing $\mathbf{K} = \begin{bmatrix} \mathbf{J} + \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix}$, we have that

$$\boldsymbol{\eta} = \sum_{k\in\Delta} \boldsymbol{\beta}_{1,k} - \mathbf{K}^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}} \end{pmatrix} = \boldsymbol{b} + \gamma^* \cdot \mathbf{K}_2^\top \mathbf{G} + \boldsymbol{e}^*$$

for some error $\boldsymbol{e}^*$. Hence, with negligible probability all first $m-1$ coefficients of $\boldsymbol{\eta}$ will be smaller than $q/4$.

B.2 Parameter Setting of (Q, poly)-PHPE in Section 5.2

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 $\mathsf{TrapGen}$ in Theorem 2.13.
3. We must choose N, Q, v such that Lemma 5.4 holds.
4. We must choose s_B such that LWE assumption holds.
5. We set s used in $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ such that the output matrices are statistically indistinguishable.
6. We set s such that Lemma 2.7 holds.
7. We set s and ρ to meet the requirements in Theorem 4.2.
8. We set s_D such that $\boldsymbol{e}'_i \stackrel{s}{\approx} (\mathbf{J}^*_i)^\top \cdot \boldsymbol{e} + \boldsymbol{e}'_i$.

Next, we choose one setting of parameters to demonstrate feasibility under the constraints. Particularly, we can choose $n = \mathsf{poly}(\lambda)$, $Q = O(\lambda)$, $\kappa = \omega(\log \lambda)$, $v = O(\kappa)$, $N = O(Q\kappa^3)$, $m = O(n \log q)$, $s_B = \sqrt{n}$, $s = O(\ell n \log q)^{O(d)} \cdot \omega(\sqrt{\log m})$, $\rho = s\sqrt{m} \cdot O(\ell n \log q)^{O(d)} \cdot \lambda^{\omega(1)}$, $s_D = \rho\sqrt{mn} \cdot \lambda^{\omega(1)}$, $q = \rho v m \sqrt{n} \cdot \lambda^{\omega(1)}$.

B.3 Security of (Q, poly)-PHPE in Section 5.2

Theorem (Restatement of Theorem 5.5) *Assuming the hardness of* LWE*, then the* QPHPE *scheme described in Section 5.2 is* (Q, poly)-Sel-Sim *secure that allows both pre- and post-challenge* 1*-key queries up to* Q *times and* 0*-key queries for an unbounded polynomial times, as Definition 2.17.*

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. This suffices to prove the theorem.

Simulator $\mathsf{Sim}(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|}, b, \mathsf{st})$:

1. $\mathsf{Sim}_1(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all public parameters as in the real $\mathsf{PH.Setup}$, except that it runs $(\mathbf{A}', \mathbf{T}_{\mathbf{A}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$, then parses $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \boldsymbol{z}^\top \end{bmatrix}$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and sets $\mathbf{A}$ be the public matrix in $\mathsf{PH.mpk}$.
2. $\mathsf{Sim}_2(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: It generates all keys using the real $\mathsf{PH.KeyGen}$.
3. $\mathsf{Sim}_3(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|}, b, \mathsf{List})$: It takes as input the public attributes $\boldsymbol{y}$, the size of the private attributes $\boldsymbol{x}$, the message b, and a list List. It constructs the challenge ciphertext as follows.
   - It samples $\boldsymbol{u}_i, \boldsymbol{v}_j$ independently and uniformly from $\mathbb{Z}_q^m$, and sets $\boldsymbol{\beta}_0 = \boldsymbol{z}$, where $\boldsymbol{z}$ is the vector prepared in Sim_1.
   - If $(b, \mathsf{List}) = \perp$, it computes $\{\boldsymbol{\beta}_{1,k}\}_{k \in [N]}$ as follows:
     - Sample random vectors $\widetilde{\boldsymbol{\beta}}_k$ from $\mathbb{Z}_q^m$ for $k \in [N]$.

- Choose $2Q$ random subsets $\Delta_1, \cdots, \Delta_Q, \Delta'_1, \cdots, \Delta'_Q$ of $[N]$ according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, each of which has cardinality v. Note that with an overwhelming probability, the $2Q$ subsets would be cover-free under our parameter selection.
- Generate random shares $\{b_k\}_{k\in[N]}$ over $\mathbb{Z}_q$ under the following constraints: for $\hat{i} \in [Q]$, (1) $\sum_{k\in\Delta_{\hat{i}}} b_k = 0$, and (2) $\sum_{k\in\Delta'_{\hat{i}}} b_k = \lceil q/2 \rceil$. This can be done efficiently by the cover-freeness of the subsets, using the following standard procedure.
  First, let $\delta_{\hat{i}}$ be a unique index that only appears in $\Delta_{\hat{i}}$ but not the other subsets, and $\delta'_{\hat{i}}$ be a unique index of $\Delta'_{\hat{i}}$. To generate the random shares $\{b_k\}_{k\in[N]}$, we first sample b_k randomly for all $k \in [N] \setminus (\{\delta_{\hat{i}}\}_{\hat{i}\in[Q]} \cup \{\delta'_{\hat{i}}\}_{\hat{i}\in[Q]})$, and then fix $b_{\delta_{\hat{i}}} = -\sum_{k\in\Delta_{\hat{i}}\setminus\{\delta_{\hat{i}}\}} b_k$ for $\hat{i} \in [Q]$, and similarly $b_{\delta'_{\hat{i}}} = \lceil q/2 \rceil - \sum_{k\in\Delta'_{\hat{i}}\setminus\{\delta'_{\hat{i}}\}} b_k$ for $\hat{i} \in [Q]$.
- Set $\boldsymbol{b}_k = [0, \cdots, 0, b_k] \in \mathbb{Z}_q^m$ for $k \in [N]$, and sample errors $\{\boldsymbol{e}'_k\}_{k\in[N]}$ from the distribution $\mathcal{D}_{\mathbb{Z}_q^m, s_D}$.
- Set $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}'_k$ for $k \in [N]$.

– If $b = \mu$ and $\mathsf{List} = \{\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}\}_{\hat{i}\in[Q']}$ for some $Q' \le Q$, it computes the simulated ciphertext as follows.
- For $\hat{i} \in [Q']$, compute $\boldsymbol{u}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP}, \boldsymbol{y}\big)$, and let $\left(\Delta_{\hat{i}}, \mathbf{K}^*_{\hat{i}} = \begin{bmatrix} \mathbf{J}^*_{\hat{i}} + \mathbf{K}^*_{\hat{i},1} \\ \mathbf{K}^*_{\hat{i},2} \end{bmatrix}\right)$ be the keys for $\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}$, generated by Sim_2 for the pre-challenge 1-key queries.
- Sample $Q - Q'$ random subsets of cardinality v according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, i.e., $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q'+1,Q]}$, starting with the index $Q'+1$ and ending with Q. We know that by our setting of parameters, the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ are cover-free with an overwhelming probability.
- Compute vectors $\{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}$ as follows:
  * Sample random shares $\{\mu_k\}_{k\in[N]}$ conditioned that $\sum_{k\in\Delta_{\hat{i}}} \mu_k = \lceil q/2 \rceil \mu$ for $\hat{i} \in [Q]$. Then set $\boldsymbol{b}_k = [0, \cdots, 0, \mu_k]$ for $k \in [N]$.
  * Sample random vectors $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ condition on the following equations:

$$\sum_{k\in\Delta_{\hat{i}}} \widetilde{\boldsymbol{\beta}}_k = \begin{bmatrix} \mathbf{J}^*_{\hat{i}} + \mathbf{K}^*_{\hat{i},1} \\ \mathbf{K}^*_{\hat{i},2} \end{bmatrix}^\top \cdot \begin{pmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{u}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}} \end{pmatrix} \text{ for } \hat{i} \in [Q'].$$

    The above two steps can be done efficiently due to the cover-freeness of the subsets $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$. The procedure is the same as we have presented in the previous case.
  * Sample errors $\{\boldsymbol{e}_k\}_{k\in[N]}$ according $\mathcal{D}_{\mathbb{Z}_q^m, s_D}$.
  * Set $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}'_k$ for $k \in [N]$.

– It outputs the challenge ciphertext

$$\mathsf{ct}^* = \big(\{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}, \boldsymbol{y}, \boldsymbol{\beta}_0, \{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}\big).$$

4. $\mathsf{Sim}_4(1^\lambda, \boldsymbol{y}, 1^{|\boldsymbol{x}|})$: If the query is a 0-key, then it generates the key using the real $\mathsf{QPH.KeyGen}$. Otherwise, we denote function $\widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}$ be the adversary's 1-key query and $(\mu, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP}_{\gamma_{\hat{i}}})$ be the message received from the oracle $\mathcal{O}$. Here we use index $\hat{i} \in [Q]$ to denote the number of overall 1-key queries up to this point. Then the simulator computes as follows.
   - The simulator first considers the following two cases to determine the parameter Δ:
     - Case 1: $Q' = 0$, i.e., the adversary did not make any 1-key pre-challenge query.
       * If $\mu = 0$, set $\Delta := \Delta_{\hat{i}}$.
       * Else $\Delta := \Delta'_{\hat{i}}$, where $\{\Delta_{\hat{i}}\}_{\hat{i}\in[Q]}$ and $\{\Delta'_{\hat{i}}\}_{\hat{i}\in[Q]}$ are the subsets prepared by Sim_3 in the previous procedure.
     - Case 2: $1 \le Q' < Q$, i.e., the adversary had made Q' 1-key pre-challenge queries.
       * Set $\Delta := \Delta_{\hat{i}}$ where $\Delta_{\hat{i}}$ is the subset prepared by Sim_3 (where μ had been received by Sim_3) in the previous procedure.
   - Compute $\mathbf{P}^*_\Delta = \sum_{k\in\Delta} \mathbf{P}_k$, and compute $\widetilde{\boldsymbol{\beta}}_\Delta = \sum_{k\in\Delta} \widetilde{\boldsymbol{\beta}}_k$, where $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ are the vectors prepared by Sim_3 in the previous procedure.
   - Compute $\mathbf{A}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{pk}}(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP})$, and compute $\boldsymbol{u}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \widehat{C^*_{\hat{i}}} \circ \mathsf{IP}, \boldsymbol{y}\big)$.
   - Sample $\mathbf{J}^*_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, and use $\mathbf{T}_{\mathbf{A}'}$ to sample $\begin{bmatrix}\mathbf{K}^*_{\hat{i},1}\\ \mathbf{K}^*_{\hat{i},2}\end{bmatrix} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times m},s}$ such that
   $$\begin{bmatrix}\mathbf{A} & \mathbf{A}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}}\\ \boldsymbol{\beta}_0^\top & \boldsymbol{u}^\top_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{K}^*_{\hat{i},1}\\ \mathbf{K}^*_{\hat{i},2}\end{bmatrix} = -\begin{bmatrix}\mathbf{A}\\ \boldsymbol{\beta}_0^\top\end{bmatrix} \cdot \mathbf{J}^*_{\hat{i}} + \begin{bmatrix}\mathbf{P}^*_\Delta\\ \widetilde{\boldsymbol{\beta}}^\top_\Delta\end{bmatrix}.$$
   - Output $\mathsf{sk}_{\widehat{C^*_{\hat{i}}}\circ\mathsf{IP}_{\gamma_{\hat{i}}}} = \Big(\Delta, \begin{bmatrix}\mathbf{J}^*_{\hat{i}} + \mathbf{K}^*_{\hat{i},1}\\ \mathbf{K}^*_{\hat{i},2}\end{bmatrix}\Big)$.

Auxiliary Algorithms.

$\mathsf{QPH.Setup}^*_1(1^\lambda, 1^d, 1^Q, \boldsymbol{x}, \boldsymbol{y})$: Do the following:

1. Sample a random matrix with associated trapdoor:
   $$(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q).$$
2. Let $\mathbf{A}_i = \mathbf{A}\cdot\mathbf{R}_i - y_i\cdot\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ for $i \in [\ell]$, where $\mathbf{R}_i \xleftarrow{\$} \{-1,1\}^{m\times m}$ for $i \in [\ell]$.
3. Let $\mathbf{B}_j = \mathbf{A}\cdot\mathbf{R}'_j - x_j\cdot\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ for $j \in [t]$, where $\mathbf{R}'_j \xleftarrow{\$} \{-1,1\}^{m\times m}$ for $j \in [t]$.
4. Choose Q random subsets $\Delta^*_1, \cdots, \Delta^*_Q$ with size v according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$. 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.

5. 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 random matrices $\mathbf{P}_k$ from $\mathbb{Z}_q^{m\times n}$ for $k \in [N]$ under the constraint $\sum_{k\in\Delta_{\hat{i}}^*} \mathbf{P}_k = \mathbf{A}\cdot\mathbf{J}_{\hat{i}}^*$, and we denote $\sum_{k\in\Delta_{\hat{i}}^*} \mathbf{P}_k$ as $\mathbf{P}_{\Delta_{\hat{i}}^*}$.
6. Output the master public key as $\mathsf{PH.mpk} = \big(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \mathbf{A}, \{\mathbf{P}_k\}_{k\in[N]}\big)$ and the master secret key as

$$\mathsf{PH.msk} = \big(\mathbf{T}_{\mathbf{A}}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_j^{'}\}_{j\in[t]}, \{\Delta_{\hat{i}}^*, \mathbf{J}_{\hat{i}}^*\}_{\hat{i}\in[Q]}\big).$$

$\mathsf{QPH.Enc}_1^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: The ciphertext is computed as follows:

1. Sample $\boldsymbol{s}, \boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^n, s_B}$ and set $\boldsymbol{\beta}_0 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}$.
2. For $i \in [\ell], j \in [t]$ compute $\boldsymbol{u}_i = \mathbf{R}_i^\top \cdot \boldsymbol{\beta}_0, \quad \boldsymbol{v}_j = (\mathbf{R}_j^{'})^\top \cdot \boldsymbol{\beta}_0$, where the matrices $\{\mathbf{R}_i^\top\}$ and $\{\mathbf{R}_j^{'}\}$ are the matrices in the msk generated by $\mathsf{PH.Setup}_1^*$.
3. $\{\boldsymbol{\beta}_{1,k}\}$ is computed as real encryption algorithm.
4. Output $\big(\{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}, \boldsymbol{y}, \boldsymbol{\beta}_0, \{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}\big)$.

$\mathsf{QPH.KeyGen}_1^*(\mathsf{QPH.msk}, \widehat{C}\circ\mathsf{IP}_\gamma)$: This algorithm is stateful that keeps track of how many keys have been queried before. Particularly, it does the following:

1. Compute the homomorphic public key corresponding to circuit $\widehat{C}\circ\mathsf{IP}$ as

$$\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{pk}}\big(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \widehat{C}\circ\mathsf{IP}\big)$$

2. We know that $\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} = \mathbf{A}\cdot\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} - \langle\mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\rangle\cdot\mathbf{G}$. For different types of the key query, we respond as follows:
   - 0-key query $\widehat{C}\circ\mathsf{IP}_\gamma$ such that $\widehat{C}\circ\mathsf{IP}(\boldsymbol{x},\boldsymbol{y}) \neq \gamma$. In this case, we have:

$$\big[\mathbf{A}|\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\cdot\mathbf{G}\big] = [\mathbf{A}|\mathbf{A}\cdot\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} + (\gamma - \langle\mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\rangle)\cdot\mathbf{G}\big].$$

     As the $\mathbf{G}$ trapdoor does not vanish, the algorithm samples a fresh random subset $\Delta \subseteq [N]$ with cardinality v according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, and then runs

$$\mathbf{K} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \mathbf{R}_{\widehat{C}\circ\mathsf{IP}}, \gamma - \langle\mathsf{PT}(\boldsymbol{x}), \widehat{C}(\boldsymbol{y})\rangle, \sum_{k\in\Delta}\mathbf{P}_k, s),$$

     satisfying $\big[\mathbf{A}|\mathbf{A}\cdot\mathbf{R}_{\widehat{C}\circ\mathsf{IP}} + \gamma\cdot\mathbf{G}\big]\cdot\mathbf{K} = \sum_{k\in\Delta}\mathbf{P}_k$.
   - 1-key query of function $\widehat{C}\circ\mathsf{IP}_\gamma$ such that $\widehat{C}\circ\mathsf{IP}(\boldsymbol{x},\boldsymbol{y}) = \gamma$. In this case, the algorithm does the following:
     - Let $\widehat{C}_{\hat{i}}\circ\mathsf{IP}_{\gamma_{\hat{i}}}$ be the $\hat{i}$-th 1-key query, where $\hat{i} \in [Q]$. Then set $\Delta = \Delta_{\hat{i}}^*$ instead of sampling freshly. We recall that $\Delta_{\hat{i}}^*$ is the subset sampled in the $\mathsf{QPH.Setup}_1^*$, and $\mathbf{J}_{\hat{i}}^*$ is the corresponding matrix.
     - Compute

$$\mathbf{R}_{\widehat{C}_{\hat{i}}\circ\mathsf{IP}} = \mathsf{Eval}_{\mathsf{trap}}\big(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_j^{'}\}_{j\in[t]}, \mathbf{A}, \widehat{C}_{\hat{i}}\circ\mathsf{IP}, \boldsymbol{x}, \boldsymbol{y}\big)$$

- Sample $\mathbf{K}_{\hat{i},2} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$, and set $\mathbf{K} = \begin{pmatrix} \mathbf{J}_{\hat{i}}^* - \mathbf{R}_{\widehat{C}_{\hat{i}}^* \circ \mathsf{IP}} \cdot \mathbf{K}_{\hat{i},2} \\ \mathbf{K}_{\hat{i},2} \end{pmatrix}$.

  Note that by construction, we have

$$\big[\mathbf{A}|\mathbf{A}_{\widehat{C}_{\hat{i}} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}\big] \cdot \mathbf{K} = \big[\mathbf{A}|\mathbf{A} \cdot \mathbf{R}_{\widehat{C}_{\hat{i}} \circ \mathsf{IP}}\big] \cdot \mathbf{K} = \sum_{k \in \Delta} \mathbf{P}_k.$$

3. Return $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma} = (\Delta, \mathbf{K})$.

$\mathsf{QPH.Setup}_2^*(1^\lambda, 1^d, 1^Q, \boldsymbol{x}, \boldsymbol{y})$: Do the following:

1. For every $\hat{i} \in [Q]$, choose a unique index $\delta_{\hat{i}}$ that only appears in $\Delta_{\hat{i}}^*$ but not the other subsets. This choice exists due to the cover-freeness of the subsets, which happens with an overwhelming probability.
2. Sample $\mathbf{J}_k \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$ for $i \in [N]\backslash\{\delta_1, \cdots, \delta_Q\}$. Set $\mathbf{J}_{\delta_{\hat{i}}} = \mathbf{J}_{\hat{i}}^* - \sum_{k \in \Delta_{\hat{i}}^* \backslash \{\delta_{\hat{i}}\}} \mathbf{J}_k$ for $\hat{i} \in [Q]$.
3. Sample $\mathbf{A}$ randomly and set $\mathbf{P}_i = \mathbf{A} \cdot \mathbf{J}_i$ for $i \in [N]$.
4. The remaining elements are defined the same as in $\mathsf{QPH.Setup}_1^*$.
5. Output the master public key as $\mathsf{PH.mpk} = \big(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{B}_j\}_{j\in[t]}, \mathbf{A}, \{\mathbf{P}_k\}_{k\in[N]}\big)$ and the master secret key as

$$\mathsf{PH.msk} = \big(\mathbf{T}_{\mathbf{A}}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_j^{'}\}_{j\in[t]}, \{\Delta_{\hat{i}}^*, \mathbf{J}_{\hat{i}}^*, \delta_{\hat{i}}\}_{\hat{i}\in[Q]}, \{\mathbf{J}_k\}_{k\in[N]}\big).$$

$\mathsf{QPH.Enc}_2^*(\mathsf{QPH.mpk}, \mathsf{QPH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: The ciphertext is computed as:

1. Generate $\boldsymbol{\beta}_0, \{\boldsymbol{u}_i\}, \{\boldsymbol{v}_j\}$ as $\mathsf{PH.Enc}_1^*$.
2. Compute $\{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}$ as follows.
   - Sample $\boldsymbol{e}_k' \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}, k \in [N]$.
   - Set $\boldsymbol{\beta}_{1,k} = (\mathbf{J}_k)^\top \cdot \boldsymbol{\beta}_0 + \boldsymbol{e}_k' + \boldsymbol{b}_k$, where $\boldsymbol{b}_k = [0 \cdots 0, \frac{\lceil q/2 \rceil}{v}\mu]^\top \in \mathbb{Z}_q^m$ as defined in the real encryption algorithm.
3. Output $\big(\boldsymbol{y}, \{\boldsymbol{u}_i\}_{i\in[\ell]}, \{\boldsymbol{v}_j\}_{j\in[t]}, \boldsymbol{\beta}_0, \{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}\big)$.

$\mathsf{QPH.Enc}_3^*(\mathsf{QPH.mpk}, \mathsf{QPH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: Sample $\boldsymbol{\beta}_0 \leftarrow \mathbb{Z}_q^m$ randomly. Compute the remaining ciphertext elements as in $\mathsf{PH.Enc}_2^*$.

$\mathsf{QPH.KeyGen}_2^*(\mathsf{QPH.msk}, \widehat{C} \circ \mathsf{IP}_\gamma)$: Do the following:

1. If $\widehat{C} \circ \mathsf{IP}_\gamma$ is a 0-key query, sample the key $\mathbf{K}$ using the $\mathsf{SampleLeft}$ algorithm as in KeyGen.
2. If $\widehat{C} \circ \mathsf{IP}_\gamma$ is a 1-key query, set $\mathbf{K}$ the same as $\mathsf{PH.KeyGen}_1^*$.
3. Return $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma} = (\Delta, \mathbf{K})$.

$\mathsf{QPH.Setup}_3^*(1^\lambda, 1^d, 1^Q, \boldsymbol{x}, \boldsymbol{y})$: Do the following:

1. Sample $(\mathbf{A}', \mathbf{T}_{\mathbf{A}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$, and then parse $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \boldsymbol{z}^\top \end{bmatrix}$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \boldsymbol{z} \in \mathbb{Z}_q^m$.
2. Define $\widetilde{\boldsymbol{\beta}}_k = (\mathbf{J}_k)^\top \cdot \boldsymbol{z}$, for $k \in [N]$.
3. Set $\mathbf{A}$ as the public matrix in $\mathsf{QPH.mpk}$, and additionally store $\boldsymbol{z} \in \mathbb{Z}_q^m$ and $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ in $\mathsf{QPH.msk}$.
4. The remaining elements are defined the same as in $\mathsf{QPH.Setup}_2^*$.

$\mathsf{QPH.Enc}_4^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: Compute ciphertext elements the same as $\mathsf{QPH.Enc}_3^*$ except for setting $\boldsymbol{\beta}_0 = \boldsymbol{z}$, and $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}_k'$ for $k \in [N]$.

$\mathsf{QPH.Setup}_4^*(1^\lambda, 1^d, 1^Q, \boldsymbol{x}, \boldsymbol{y})$: Do the following:

1. Generate public matrices $\{\mathbf{P}_k\}_{k\in[N]}$ by sampling random matrices $\mathbf{P}_k$ from $\mathbb{Z}_q^{m\times n}$ for $k \in [N]$ under the constraint $\sum_{k\in\Delta_{\hat{i}}^*} \mathbf{P}_k = \mathbf{A}\cdot\mathbf{J}_{\hat{i}}^*$, which is the same as $\mathsf{QPH.Setup}_1^*$.
2. Additionally, sample random vectors $\widetilde{\boldsymbol{\beta}}_k$ from $\mathbb{Z}_q^m$ for $k \in [N]$ under the constraint $\sum_{k\in\Delta_{\hat{i}}^*} \widetilde{\boldsymbol{\beta}}_k = (\mathbf{J}_{\hat{i}}^*)^\top \cdot \boldsymbol{z}$, and denote $\sum_{k\in\Delta_{\hat{i}}^*} \widetilde{\boldsymbol{\beta}}_k$ as $\widetilde{\boldsymbol{\beta}}_{\Delta_{\hat{i}}^*}$.
3. The remaining elements are defined the same as in $\mathsf{QPH.Setup}_3^*$.

$\mathsf{QPH.KeyGen}_3^*(\mathsf{PH.msk}, \widehat{C}\circ\mathsf{IP}_\gamma)$: Do the following:

1. If $\widehat{C}\circ\mathsf{IP}_\gamma$ is a 0-key query, sample the key $(\Delta, \mathbf{K})$ the same as in $\mathsf{QPH.KeyGen}_2^*$.
2. If $\widehat{C}\circ\mathsf{IP}_\gamma$ is the $\hat{i}$-th 1-key query with $\hat{i} \in [Q]$, then set $\Delta = \Delta_{\hat{i}}^*$, the subset prepared in $\mathsf{QPH.Setup}_1^*$. Then compute $\mathbf{K}$ as follows: sample a fresh $\mathbf{J}_{\hat{i}}^{*'} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$ and use $\mathbf{T}_{\mathbf{A}'}$ to sample $\begin{bmatrix}\mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$ by $\mathsf{SampleLeft}$ algorithm such that

$$\begin{bmatrix}\mathbf{A} & \mathbf{A}_{\widehat{C}\circ\mathsf{IP}}\\ \boldsymbol{z}^\top & \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}^\top\end{bmatrix}\cdot\begin{bmatrix}\mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix} = -\begin{bmatrix}\mathbf{A}\\ \boldsymbol{z}^\top\end{bmatrix}\cdot\mathbf{J}_{\hat{i}}^{*'} + \begin{bmatrix}\mathbf{P}_\Delta\\ \widetilde{\boldsymbol{\beta}}_\Delta^\top\end{bmatrix}.$$

3. Set $\mathbf{K} = \begin{bmatrix}\mathbf{J}_{\hat{i}}^{*'} + \mathbf{K}_{\hat{i},1}\\ \mathbf{K}_{\hat{i},2}\end{bmatrix}$, and return $\mathsf{sk}_{\widehat{C}\circ\mathsf{IP}_\gamma} = (\Delta, \mathbf{K})$.

$\mathsf{QPH.Setup}_5^*(1^\lambda, 1^d, 1^Q, \boldsymbol{x}, \boldsymbol{y})$: Sample $\{\mathbf{A}_i\}, \{\mathbf{B}_j\}, \{\mathbf{P}_k\}$ the same as the real setup algorithm, and set $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ to be uniformly random. Everything else remains the same as $\mathsf{QPH.Setup}_4^*$.

$\mathsf{QPH.Enc}_5^*(\mathsf{PH.mpk}, \mathsf{PH.msk}, \boldsymbol{y}, \mu, \mathsf{List})$: the algorithm samples $\boldsymbol{u}_i, \boldsymbol{v}_j$ randomly, and then considers the following two cases:

- Case 1: the adversary has never made a pre-challenge 1-key query, i.e., $Q' = 0$. In this case, the algorithm samples Q (new) random subsets $\Delta_1', \cdots, \Delta_Q'$ of $[N]$ of cardinality v according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, and then generates random shares $\{b_k\}_{k\in[N]}$ over $\mathbb{Z}_q$ under the following constraints: for $\hat{i} \in [Q]$, (1) $\sum_{k\in\Delta_{\hat{i}}^*} b_k = \lceil q/2\rceil\mu$, and (2) $\sum_{k\in\Delta_{\hat{i}}'} b_k = \lceil q/2\rceil(1-\mu)$. Then set $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{e}_k' + \boldsymbol{b}_k$, where $\boldsymbol{e}_k' \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}$ for $k \in [N]$ and $\boldsymbol{b}_k = [0\cdots0, b_k]^\top \in \mathbb{Z}_q^m$. Everything else remains the same as $\mathsf{QPH.Enc}_4^*$.
- Case 2: the adversary has made at least one pre-challenge 1-key query, i.e., $Q' \geq 1$. The algorithm samples random shares $\{\mu_k\}_{k\in[N]}$ conditioned that $\sum_{k\in\Delta_{\hat{i}}^*} \mu_k = \lceil q/2\rceil\mu$ for $\hat{i} \in [Q]$. Then set $\boldsymbol{b}_k = [0, \cdots, 0, \mu_k]$ and $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{e}_k' + \boldsymbol{b}_k$, where $\boldsymbol{e}_k' \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_D}$ for $k \in [N]$. Everything else remains the same as $\mathsf{QPH.Enc}_4^*$.

Now we establish the proof via the following hybrids, where the first one corresponds to the real world and the last corresponds to the ideal world.

Hybrids.

Hybrid 0: The real experiment.

Hybrid 1: The real game algorithms $\mathsf{QPH.Setup}$ and $\mathsf{QPH.Enc}$ are replaced with $\mathsf{QPH.Setup}_1^*$ and $\mathsf{QPH.Enc}_1^*$, which use the knowledge of $(\boldsymbol{x}, \boldsymbol{y})$ to generate the public parameters, the master keys.

Hybrid 2: The real game $\mathsf{QPH.KeyGen}$ is replaced with $\mathsf{QPH.KeyGen}_1^*$ where instead of using the trapdoor $\mathbf{T}_{\mathbf{A}}$ of the matrix $\mathbf{A}$, the secret keys for 0-key queries are sampled using the public trapdoor $\mathbf{T}_{\mathbf{G}}$ along with the trapdoor information generated using $\mathsf{QPH.Setup}_1^*$, and the secret key for the i-th 1-key query $\widehat{C}_i^* \circ \mathsf{IP}_{\gamma_i}$ is generated as $\left(\Delta_i^*, \begin{bmatrix} \mathbf{J}_i^* - \mathbf{R}_{\widehat{C}_i^* \circ \mathsf{IP}} \cdot \mathbf{K}_{i,2} \\ \mathbf{K}_{i,2} \end{bmatrix}\right)$.

Hybrid 3: $\mathsf{QPH.Setup}_1^*$ is replaced by $\mathsf{QPH.Setup}_2^*$. In this hybrid, the public matrices $\{\mathbf{P}_i\}$ are generated by first sampling matrices $\{\mathbf{J}_i\}$ according Gaussian under certain constraints, then setting each $\mathbf{P}_i = \mathbf{A} \cdot \mathbf{J}_i$.

Hybrid 4: $\mathsf{QPH.Enc}_1^*$ is replaced by $\mathsf{QPH.Enc}_2^*$, where the difference is the computation of $\{\boldsymbol{\beta}_{1,k}\}$.

Hybrid 5: The encryption algorithm is changed from $\mathsf{QPH.Enc}_2^*$ to $\mathsf{QPH.Enc}_3^*$. Here, the ciphertext element $\boldsymbol{\beta}_0$ is switched to random and all other ciphertext elements are derived from it.

Hybrid 6: The algorithm $\mathsf{QPH.KeyGen}_1^*$ is replaced with $\mathsf{QPH.KeyGen}_2^*$. The $\mathsf{QPH.KeyGen}_2^*$ algorithm is as the real $\mathsf{QPH.KeyGen}$ algorithm, except for the response to 1-key queries $\widehat{C}^* \circ \mathsf{IP}_{\gamma}$.

Hybrid 7: The $\mathsf{QPH.Setup}_2^*$ algorithm is replaced with $\mathsf{QPH.Setup}_3^*$, where the differences are the generation of the public matrix $\mathbf{A}'$ together with $\mathbf{T}_{\mathbf{A}'}$, and the vectors $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ are added into the master secret key.

Hybrid 8: The encryption algorithm $\mathsf{QPH.Enc}_3^*$ is replaced with $\mathsf{QPH.Enc}_4^*$, where the differences are the computations of $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_{1,k}$ for $k \in [N]$.

Hybrid 9: The $\mathsf{QPH.Setup}_3^*$ algorithm is replaced with $\mathsf{QPH.Setup}_4^*$, where the differences are the computations of matrices $\mathbf{P}_k$ and vectors $\widetilde{\boldsymbol{\beta}}_k$.

Hybrid 10: The algorithm $\mathsf{QPH.KeyGen}_2^*$ is replaced with $\mathsf{QPH.KeyGen}_3^*$, where the keys for 1-key queries are generated by using trapdoor $\mathbf{T}_{\mathbf{A}'}$ such that

$$\begin{bmatrix} \mathbf{A} & \mathbf{A}_{\widehat{C}\circ\mathsf{IP}} \\ \boldsymbol{z}^\top & \boldsymbol{u}_{\widehat{C}\circ\mathsf{IP}}^\top \end{bmatrix} \cdot \begin{bmatrix} \mathbf{K}_{i,1} \\ \mathbf{K}_{i,2} \end{bmatrix} = - \begin{bmatrix} \mathbf{A} \\ \boldsymbol{z}^\top \end{bmatrix} \cdot \mathbf{J}_i^{*'} + \begin{bmatrix} \mathbf{P}_\Delta \\ \widetilde{\boldsymbol{\beta}}_\Delta^\top \end{bmatrix}.$$

Hybrid 11: The algorithm $\mathsf{QPH.Setup}_4^*$ is replaced by $\mathsf{QPH.Setup}_5^*$ algorithm. Here the public matrices $\mathbf{A}_i, \mathbf{B}_j, \mathbf{P}_k$ are generated as the real world, and $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ are sampled uniformly at random.

Hybrid 12: The encryption algorithm is changed from $\mathsf{QPH.Enc}_4^*$ to $\mathsf{QPH.Enc}_5^*$, in which the secret sharing procedure is generated differently.

Hybrid 13: The ideal experiment.

Below we show that each two adjacent hybrids are indistinguishable.

Lemma B.1 Hybrid 0 *and* Hybrid 1 *are statistically indistinguishable.*

Proof. The difference between the two hybrids is in how the public parameters and the ciphertext are generated. In Hybrid 0, for QPH.mpk we have

$$\{\mathbf{A}_i\}, \{\mathbf{B}_j\}, \{\mathbf{P}_k\} \xleftarrow{\$} \mathbb{Z}_q^{n\times m} \quad \forall i \in [\ell], j \in [t], k \in [N],$$

and for ct^*,

$$\boldsymbol{\beta}_0 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}, \boldsymbol{u}_i = (\mathbf{A}_i + y_i \cdot \mathbf{G})^\top \boldsymbol{s} + \mathbf{R}_i^\top \boldsymbol{e}, \boldsymbol{v}_j = (\mathbf{B}_j + x_j \cdot \mathbf{G})^\top \boldsymbol{s} + (\mathbf{R}'_j)^\top \boldsymbol{e}.$$

In Hybrid 1, for QPH.mpk,

$$\{\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - y_i \cdot \mathbf{G}\}, \{\mathbf{B}_j = \mathbf{A} \cdot \mathbf{R}_j^{'} - x_j \cdot \mathbf{G}\}, \quad \forall i \in [\ell], j \in [t],$$

and $\{\mathbf{P}_k\}_{k\in[N]}$ are sampled randomly under the condition $\sum_{k\in\Delta_{\hat{i}}^*} \mathbf{P}_k = \mathbf{A} \cdot \mathbf{J}_{\hat{i}}^*$ for $\hat{i} \in [Q]$, and for ct^*, we have $\boldsymbol{\beta}_0 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}, \boldsymbol{u}_i = \mathbf{R}_i^\top \cdot \boldsymbol{\beta}_0, \boldsymbol{v}_j = (\mathbf{R}'_j)^\top \boldsymbol{\beta}_0$.

We argue that the joint distribution of the public parameters and ciphertext

$$(\mathbf{A}, \{\mathbf{A}_i, \boldsymbol{u}_i\}_{i\in[\ell]}, \{\mathbf{B}_j, \boldsymbol{v}_j\}_{j\in[t]}, \{\mathbf{P}_k\}_{k\in[N]}, \mathsf{ct}^*)$$

is statistically indistinguishable between these two hybrids.

On the one hand, by Theorem 2.13, we have that $\mathbf{A} \overset{\$}{\approx} \mathcal{U}$ for $\mathbf{A}$ output by TrapGen algorithm. Then By Lemma 2.16, we have that

$$(\mathbf{A}, \mathbf{A} \cdot \mathbf{R}_i - y_i \cdot \mathbf{G}, \mathbf{R}_i^\top \boldsymbol{e}) \overset{s}{\approx} (\mathbf{A}, \mathcal{U}, \mathbf{R}_i^\top \boldsymbol{e}),$$

and this holds for $\mathbf{B}_j = \mathbf{A} \cdot \mathbf{R}'_j - x_j \cdot \mathbf{G}$ as well. Since for all i, $\mathbf{R}_i$ (resp. $\mathbf{R}'_j$) is randomly and independently chosen, it follows that

$$(\mathbf{A}, \{\mathbf{A}\cdot\mathbf{R}_i - y_i\cdot\mathbf{G}\}, \{\mathbf{R}_i^\top \boldsymbol{e}\}, \{\mathbf{A}\cdot\mathbf{R}'_j - x_j\cdot\mathbf{G}\}, \{(\mathbf{R}'_j)^\top \boldsymbol{e}\}) \overset{s}{\approx} (\mathbf{A}, \{\mathcal{U}_i\}, \{\mathbf{R}_i^\top \boldsymbol{e}\}, \{\mathcal{U}_j\}, \{(\mathbf{R}'_j)^\top \boldsymbol{e}\}).$$

On the other hand, by Lemma 2.7, We have $(\mathbf{A}, \{\mathbf{A}\cdot\mathbf{J}_{\hat{i}}^*\}_{\hat{i}\in[Q]}) \overset{s}{\approx} (\mathbf{A}, \{\mathcal{U}_{\hat{i}}\}_{\hat{i}\in[Q]})$ for random matrix $\mathbf{A}$. Furthermore, the ciphertext components $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are derived simply by adding $(\mathbf{A}_i + y_i \cdot \mathbf{G})^\top \cdot \boldsymbol{s}$ and $(\mathbf{B}_j + x_j \cdot \mathbf{G})^\top \cdot \boldsymbol{s}$ to $\mathbf{R}_i^\top \boldsymbol{e}$ and $(\mathbf{R}'_j)^\top \boldsymbol{e}$, respectively. As statistical distance does not increase under application of a function, this implies that the public parameters and the ciphertext are statistically close in these two hybrids. □

Lemma B.2 Hybrid 1 *and* Hybrid 2 *are statistically indistinguishable.*

Proof. Between Hybrid 1 and Hybrid 2, the manner of generating keys changes from QPH.KeyGen to $\mathsf{QPH.KeyGen}_1^*$. We consider two cases:

1. For a 0-key query $\widehat{C} \circ \mathsf{IP}_\gamma$, in Hybrid 1 the key is sampled using the SampleLeft algorithm, whereas in Hybrid 2, it is sampled using the SampleRight algorithm. By Lemma 2.15 and our setting of parameters, the resulting distributions are statistically indistinguishable.

2. For a 1-key query $\widehat{C}\circ\mathsf{IP}_\gamma$, we note that $\{\mathbf{P}_k\}$ are chosen the same way in both Hybrid 1 and Hybrid 2. In Hybrid 1, for $\hat{i}\in[Q]$, we sample $\mathbf{K}_{\hat{i}}=\begin{bmatrix}\mathbf{J}^*_{\hat{i}}+\mathbf{K}_{1,\hat{i}}\\ \mathbf{K}_{2,\hat{i}}\end{bmatrix}$ by first sampling $\mathbf{J}^*_{\hat{i}}\leftarrow\mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$ then using SampleLeft to sample the key such that

$$\big[\mathbf{A}|\mathbf{A}_{\widehat{C}^*_{\hat{i}}\circ\mathsf{IP}}+\gamma\cdot\mathbf{G}\big]\cdot\begin{bmatrix}\mathbf{K}_{1,\hat{i}}\\ \mathbf{K}_{2,\hat{i}}\end{bmatrix}=\sum_{k\in\Delta^*_{\hat{i}}}\mathbf{P}_k-\mathbf{A}\mathbf{J}^*_{\hat{i}},$$

where the marginal distribution of $\sum_{k\in\Delta^*_{\hat{i}}}\mathbf{P}_k=\mathbf{A}\cdot\mathbf{J}^*_{\hat{i}}$ is uniformly at random, as $\mathbf{J}^*_{\hat{i}}$ is hidden in the view of adversary in this case.

In Hybrid 2, for $\hat{i}\in[Q]$, we output $\mathbf{K}_{\hat{i}}=\begin{bmatrix}\mathbf{J}^*_{\hat{i}}-\mathbf{R}_{\widehat{C}\circ\mathsf{IP}}\cdot\mathbf{K}'_{2,\hat{i}}\\ \mathbf{K}'_{2,\hat{i}}\end{bmatrix}$, where $\mathbf{J}^*_{\hat{i}}\leftarrow\mathcal{D}_{\mathbb{Z}^{m\times m},\sqrt{\rho^2+s^2}}$, $\mathbf{K}'_{2,\hat{i}}\leftarrow\mathcal{D}_{\mathbb{Z}^{m\times m},s}$ and

$$\mathbf{R}_{\widehat{C}\circ\mathsf{IP}}=\mathsf{Eval}_{\mathsf{trap}}\big(\{\mathbf{R}_i\}_{i\in[\ell]},\{\mathbf{R}'_j\}_{j\in[t]},\mathbf{A},\widehat{C}\circ\mathsf{IP},\boldsymbol{x},\boldsymbol{y}\big).$$

To rephrase in the terms as we used in Section 4, the key generation (with the public $\mathbf{A},\mathbf{P}$) in Hybrid 1 is exactly the procedure of Sampler-1, and in Hybrid 2 the Sampler-2. Therefore, by Theorem 4.2, we know that the two cases are statistically indistinguishable. Via a simple hybrid argument, we know the indistinguishability also holds for Q key queries.

The above analysis shows that the hybrids are indistinguishable for either case, and this completes the proof. □

Lemma B.3 Hybrid 2 *and* Hybrid 3 *are statistically indistinguishable.*

Proof. The only difference between the two hybrids is the way how the public matrices $\{\mathbf{P}_k\}$ are generated. In Hybrid 2, for $k\in[N]$, $\mathbf{P}_k$ is sampled uniformly at random under the constraint $\sum_{k\in\Delta^*_i}\mathbf{P}_k=\mathbf{A}\cdot\mathbf{J}^*_i$. In Hybrid 3, for $k\in[N]$, $\mathbf{P}_k=\mathbf{A}\cdot\mathbf{J}_k$, where $\mathbf{J}_k\leftarrow D_{\mathbb{Z}^{m\times m},s}$ for $k\in[N]\backslash\{\delta_1,\cdots,\delta_Q\}$, and $\mathbf{J}_{\delta_{\hat{i}}}=\mathbf{J}^*_{\hat{i}}-\sum_{k\in\Delta^*_{\hat{i}}\backslash\{\delta_{\hat{i}}\}}\mathbf{J}_k$ for $i\in[Q]$.

Clearly, $\{\mathbf{P}_k\}_{k\in[N]}$ in the two hybrids are under the same constraint. It suffices to show the marginal distribution of $\mathbf{P}_k$ is uniformly at random for $k\in[N]$ in Hybrid 3. By Lemma 2.7 and our setting of parameters, we have $(\mathbf{A},\mathbf{A}\cdot\mathbf{J}_i)\overset{\mathsf{s}}{\approx}(\mathbf{A},\mathcal{U}_i)$, for $i\in[N]$. This completes the proof. □

Lemma B.4 Hybrid 3 *and* Hybrid 4 *are statistically indistinguishable.*

Proof. The only difference between the two hybrids is the way how the ciphertext elements $\{\boldsymbol{\beta}_{1,k}\}_{k\in[N]}$ are generated. In Hybrid 3, we have $\boldsymbol{\beta}_{1,k}=\mathbf{P}_k^\top\boldsymbol{s}+\boldsymbol{e}'_k+\boldsymbol{b}_k$, for $k\in[N]$. In Hybrid 4, for $k\in[N]$, we have

$$\begin{aligned}\boldsymbol{\beta}_{1,k}&=\mathbf{J}_k^\top\cdot\boldsymbol{\beta}_0+\boldsymbol{e}'_k+\boldsymbol{b}_k\\&=(\mathbf{A}\cdot\mathbf{J}_k)^\top\boldsymbol{s}+\mathbf{J}_k^\top\cdot\boldsymbol{e}+\boldsymbol{e}'_k+\boldsymbol{b}_k\\&=\mathbf{P}_k^\top\cdot\boldsymbol{s}+\mathbf{J}_k^\top\cdot\boldsymbol{e}+\boldsymbol{e}'_k+\boldsymbol{b}_k.\end{aligned}$$

Thus, it suffices to show that $\boldsymbol{e}'_k \overset{s}{\approx} \mathbf{J}_k^\top \cdot \boldsymbol{e} + \boldsymbol{e}'_k$. By the smudging lemma, i.e., Lemma 2.9 and our setting of parameters, the two distributions are statistically close. Thus this completes the proof. □

Lemma B.5 Hybrid 4 *and* Hybrid 5 *are computationally indistinguishable assuming* LWE *is hard.*

Proof. We show how the LWE assumption can be broken given an adversary that distinguishes between Hybrid 4 and Hybrid 5. Given the LWE challenge sample $(\mathbf{A}, \boldsymbol{z})$ where $\boldsymbol{z}$ is either real or random. The reduction does as follows:

1. Run $\mathsf{QPH.Setup}_2^*$ and $\mathsf{QPH.KeyGen}_1^*$. These algorithms are used to produce the public parameters and the function keys. We note that these two algorithms can be implemented without the trapdoor of $\mathbf{A}$.
2. To produce the ciphertext, set $\boldsymbol{\beta}_0 = \boldsymbol{z}$ and generate the remaining components as in Hybrid 3.

Now if $\boldsymbol{z} = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}$, then the reduction faithfully simulates the distribution of Hybrid 4, whereas if $\boldsymbol{z}$ is random, the reduction simulates Hybrid 5. Thus the reduction has the same advantage in breaking LWE as that in distinguishing the two hybrids. This completes the proof. □

Lemma B.6 Hybrid 5 *and* Hybrid 6 *are statistical indistinguishable.*

Proof. The proof is analogous to the proof of indistinguishability between Hybrid 1 and Hybrid 2 for generating secret keys of 0-key queries. □

Lemma B.7 Hybrid 6 *and* Hybrid 7 *are statistical indistinguishable.*

Proof. The difference between the two hybrids is how the public matrix $\mathbf{A}$ and the trapdoor are generated. In Hybrid 6, $\mathbf{A}$ and trapdoor are generated by $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, q, m)$. In Hybrid 7, it first runs $(\mathbf{A}', \mathbf{T}_{\mathbf{A}'}) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$, then parses $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \boldsymbol{z}^\top \end{bmatrix}$ and sets $\mathbf{A}$ as the public matrix. By Theorem 2.13, the matrix $\mathbf{A}$ in these two cases are both statistically close to the uniform distribution, and thus they are indistinguishable for the two hybrids.

On the other hand, by Theorem 2.13, $\mathbf{T}_{\mathbf{A}'}$ is also a trapdoor of $\mathbf{A}$ for KeyGen 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. Therefore, these two hybrids are statistically indistinguishable. This completes the proof. □

Lemma B.8 Hybrid 7 *and* Hybrid 8 *are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way of generating ciphertext elements $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_{1,k}$ for $k \in [N]$. In Hybrid 7, $\boldsymbol{\beta}_0$ is sampled uniformly at random, whereas in Hybrid 8, $\boldsymbol{\beta}_0$ is set as $\boldsymbol{z}$ prepared in $\mathsf{QPH.Setup}_3^*$. By theorem 2.13, $\boldsymbol{z}$ is statistically close to uniformly random, and thus the two hybrids generate β_0 in a statistically close way.

On the other hand, $\boldsymbol{\beta}_{1,k}$ in the two hybrids are identical for $k \in [k]$ by the setting of $\widetilde{\boldsymbol{\beta}}_k$. This completes the proof. □

Lemma B.9 Hybrid 8 *and* Hybrid 9 *are statistically indistinguishable.*

Proof. The difference between the two hybrids is the way of generating the public matrices $\mathbf{P}_k$ and the elements $\widetilde{\boldsymbol{\beta}}_k$ prepared in $\mathsf{QPH.msk}$ for $k \in [N]$.

For the indistinguishability of $\mathbf{P}_k$ in the two hybrids, the proof is analogous to the proof of indistinguishability between Hybrid 2 and Hybrid 3.

For $\widetilde{\boldsymbol{\beta}}_k$, in Hybrid 8, $\widetilde{\boldsymbol{\beta}}_k = (\mathbf{J}_k)^\top \cdot \boldsymbol{z}$, where $\mathbf{J}_k \leftarrow D_{\mathbb{Z}^{m\times m},s}$ for $k \in [N]$ under constraint $\sum_{k\in\Delta_{\hat{i}}^*} \mathbf{J}_k = \mathbf{J}_{\hat{i}}^*$ with $\hat{i} \in [Q]$. By Lemma 2.7, the marginal distribution of $\widetilde{\boldsymbol{\beta}}_k$ are statistically close to uniformly random distribution for $k \in [N]$ under constraint $\sum_{k\in\Delta_{\hat{i}}^*} \widetilde{\boldsymbol{\beta}}_k = (\mathbf{J}_{\hat{i}}^*)^\top \cdot \boldsymbol{z}$, which is identical to the distribution of $\widetilde{\boldsymbol{\beta}}_k$ in Hybrid 9. Thus the two hybrids are statistically close. □

Lemma B.10 Hybrid 9 *and* Hybrid 10 *are statistically indistinguishable.*

Proof. We show the indistinguishability between the two hybrids in an information theoretical way, i.e., the view of adversary $\mathcal{A}$ in Hybrid 9 is statistically indistinguishable from that in Hybrid 10: $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid9}} \overset{s}{\approx} \mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid10}}$.

From Definition 3.2, we know that

$$\mathcal{A}^{\mathsf{View}} = \big(\mathsf{QPH.mpk}, \{\mathsf{sk}_{\widehat{C}_i\circ\mathsf{IP}_{\gamma_i}}\}_{i\in[\mathsf{poly}]}, \{\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}}\}_{\hat{i}\in[Q]}, \mathsf{ct}^*\big),$$

where $\mathsf{sk}_{\widehat{C}_i\circ\mathsf{IP}_{\gamma_i}}$ is the secret key for 0-key query $\widehat{C}_i \circ \mathsf{IP}_{\gamma_i}$, $\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}}$ is the secret key for 1-key query $\widehat{C}_{\hat{i}}^* \circ \mathsf{IP}_{\gamma_{\hat{i}}}$. The only differences between the two views come from the way how 1-keys, i.e., $\{\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}}\}_{\hat{i}\in[Q]}$ are generated.

To show the indistinguishability of the two views, we define an intermediate view $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}}$, in which the elements $(\mathbf{A}, \{\mathbf{A}_i\}, \{\mathbf{B}_j\})$ in $\mathsf{QPH.mpk}$, $\{\mathsf{sk}_{\widehat{C}_i\circ\mathsf{IP}_{\gamma_i}}\}_{i\in[\mathsf{poly}]}$, $\{\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}}\}_{\hat{i}\in[Q]}$ and $\boldsymbol{\beta}_0, \{\boldsymbol{u}_i\}, \{\boldsymbol{v}_j\}$ in ct^* are identical to $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid9}}$. The public matrices $\{\mathbf{P}_k\}_{k\in[N]}$ are generated by randomly sampling $\mathbf{P}_k$ from $\mathbb{Z}_q^{n\times m}$ under the constraint $\sum_{k\in\Delta_{\hat{i}}^*} \mathbf{P}_k = [\mathbf{A}|\mathbf{A}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}}] \cdot (\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}})$, and the elements $\{\boldsymbol{\beta}_{1,k}\}$ are generated as $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}'_k$, where $\boldsymbol{b}_k, \boldsymbol{e}'_k$ are the same as before, and $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ are sampled randomly from $\mathbb{Z}_q^m$ under the constraint $\sum_{k\in\Delta_{\hat{i}}^*} \widetilde{\boldsymbol{\beta}}_k = (\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}})^\top \cdot \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{u}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}} \end{pmatrix}$. Next we will show $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid9}} \overset{s}{\approx} \mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}}$ and $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}} \overset{s}{\approx} \mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid10}}$.

For the first part, it's easy to see $[\mathbf{A}|\mathbf{A}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}}] \cdot (\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}}) = \mathbf{A} \cdot \mathbf{J}_{\hat{i}}^*$ and $(\mathsf{sk}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}_{\gamma_{\hat{i}}}})^\top \cdot \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{u}_{\widehat{C}_{\hat{i}}^*\circ\mathsf{IP}} \end{pmatrix} = (\mathbf{J}_{\hat{i}}^*)^\top \cdot \boldsymbol{z}$ in $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}}$, meaning that $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}}$ can be viewed as a change of variables of and $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid9}}$. Thus, these two distributions are identical.

For the second part, we show it by a hybrid argument. For $i^* \in \{0, \cdots, Q\}$, we define an intermediate view $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},i^*}$ as: $\mathbf{A}, \{\mathbf{A}_i\}, \{\mathbf{B}_j\}$ in $\mathsf{QPH.mpk}$, $\{\mathsf{sk}_{\widehat{C}_i\circ\mathsf{IP}_{\gamma_i}}\}_{i\in[\mathsf{poly}]}$

are identical to $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}}$, the first i^* secret keys for 1-key queries are generated by the way of Hybrid 10, the remaining $Q-i^*$ secret keys are generated as Hybrid 9, and the related constraints of $\mathbf{P}_k$ and $\widetilde{\boldsymbol{\beta}}_k$ are corresponding to the secret keys sampled in this hybrid. It is easy to see $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},0}$ is identical to $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}}$, and $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},Q}$ is exactly $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid10}}$. It thus suffices to show $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},i^*-1} \overset{s}{\approx} \mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid9},i^*}$ for every $i^* \in \{1,\cdots,Q\}$, then implying $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9}} \overset{s}{\approx} \mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid10}}$.

We show this by reduction: if there exists an $i^* \in \{1,\cdots,Q\}$ such that we can distinguish $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},i^*-1}$ and $\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},i^*}$, then we can break the indistinguishability of Theorem 4.2. The reduction T processes as follows:

1. Given the challenge elements $\mathbf{A}', \mathbf{U}'_{i^*}$ output by Stage 1 of the Sampler. Generate QPH.mpk as follows:
   - Parse $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \boldsymbol{z}^\top \end{bmatrix}$ and denote $\mathbf{A}$ as the public matrix in QPH.mpk.
   - Use $\mathbf{A}$ to generate the public elements $\{\mathbf{A}_i\}, \{\mathbf{B}_j\}$ in QPH.mpk as Hybrid10.
   - Sample $\mathbf{J}^*_i \leftarrow D_{\mathbb{Z}^{m\times m},\rho}$, and set $\mathbf{U}'_i = \mathbf{A}' \cdot \mathbf{J}^*_i = \begin{bmatrix} \mathbf{U}_i \\ \boldsymbol{u}_i^\top \end{bmatrix}$, for $i \in [Q]\backslash\{i^*\}$. Then choose Q random subsets $\Delta^*_1, \cdots, \Delta^*_Q$ according sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, and generate $\mathbf{P}'_k = \begin{bmatrix} \mathbf{P}_k \\ \widetilde{\boldsymbol{\beta}}_k^\top \end{bmatrix}$ for $k \in [N]$ by randomly sampling $\mathbf{P}'_k$ from $\mathbb{Z}_q^{(n+1)\times m}$ under the constraint $\sum_{k\in\Delta^*_i} \mathbf{P}'_k = \mathbf{U}'_i, i \in [Q]$. Denote $\{\mathbf{P}_k\}$ as the public elements in QPH.mpk.
2. Generate secret keys $\{\mathsf{sk}_{\widehat{C}_i \circ \mathsf{IP}_{\gamma_i}}\}_{i\in[\mathsf{poly}]}$ for 0-key queries $\{\widehat{C}_i \circ \mathsf{IP}_{\gamma_i}\}_{i\in\mathsf{poly}}$ such that the distributions of $\{\mathsf{sk}_{\widehat{C}_i \circ \mathsf{IP}_{\gamma_i}}\}_{i\in[\mathsf{poly}]}$ are identical to which in Hybrid 9.
3. Generate secret keys $\{\mathsf{sk}_{\widehat{C}^*_i \circ \mathsf{IP}_{\gamma_i}}\}_{i\in[Q]}$ for 1-key queries $\{\widehat{C}^*_i \circ \mathsf{IP}_{\gamma_i}\}_{i\in[Q]}$ as follows:
   - Generate the first i^*-1 secret keys for 1-key queries $\{\widehat{C}^*_i \circ \mathsf{IP}_{\gamma_i}\}_{i\in[i^*-1]}$ such that these keys distribute as Hybrid10.
   - For the i^*-th 1-key query $\widehat{C}^*_{i^*} \circ \mathsf{IP}_{\gamma_{i^*}}$, compute $\mathbf{R}_{\widehat{C}^*_{i^*}\circ\mathsf{IP}}$ and use it to call the Stage 2 oracle of the Sampler. Denote $\mathsf{sk}_{\widehat{C}^*_{i^*}\circ\mathsf{IP}_{\gamma_{i^*}}} = (\Delta^*_{i^*}, \mathbf{Y}_{i^*})$, where $\mathbf{Y}_{i^*}$ is the response of the oracle such that $[\mathbf{A}'|\mathbf{A}' \cdot \mathbf{R}_{\widehat{C}^*_{i^*}\circ\mathsf{IP}}] \cdot \mathbf{Y}_{i^*} = \mathbf{U}'_{i^*}$.
   - Generate the last $Q-i^*$ secret keys for 1-key queries $\{\widehat{C}^*_i \circ \mathsf{IP}\}_{i\in[i^*+1,Q]}$ such that these keys distribute as Hybrid9.
4. Set the element $\boldsymbol{\beta}_0 = \boldsymbol{z}$, and compute $\{\boldsymbol{u}_i\}, \{\boldsymbol{v}_j\}$ in ct^* as Hybrid9. For $\boldsymbol{\beta}_{i,k}, k \in [N]$, set $\boldsymbol{\beta}_{1,k} = \widetilde{\boldsymbol{\beta}}_k + \boldsymbol{b}_k + \boldsymbol{e}'_k$, where $\boldsymbol{b}_k, \boldsymbol{e}'_k$ are defined as Hybrid9, and $\{\widetilde{\boldsymbol{\beta}}_k\}$ are the vectors generated in Step 1 along with the matrices $\{\mathbf{P}_k\}$.

Denote D^* as the distribution of challenge tuple $(\mathbf{A}', \mathbf{A}' \cdot \mathbf{R}_{\widehat{C}^*_{i^*}\circ\mathsf{IP}}, \mathbf{Y}_{i^*}, \mathbf{U}'_{i^*})$ of Sampler, and then we can treat the distribution of the view output by the reduction above as $T(D^*)$. Furthermore, denote D_1 as the distribution of tuple output by Sampler-1, and D_2 as that by Sampler-2. Now if $D^* = D_2$, then it is clear from the construction that $T(D_2) = \mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid9},i^*-1}$. On the other hand, if $D^* = D_1$, the distribution $T(D_1)$ however, is not exactly the same as that of

$T(D_2) = \mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid}9,i^*-1}$ due to a subtle mismatch in the anticipated distribution of $\mathbf{U}'_{i^*}$ in $\mathsf{Hybrid}9, i^*$ is $\mathbf{A} \cdot \mathbf{J}^*_i$, not exactly uniformly random as the D_1.

To handle this issue, we further define D_3 as the distribution of tuple $(\mathbf{A}', \mathbf{A}' \cdot \mathbf{R}_{\widehat{C}^*_{i^*} \circ \mathsf{IP}_{\gamma_{i^*}}}, \mathbf{Y}'_{i^*}, \mathbf{U}''_{i^*})$, where everything is the same as D_1 except that $\mathbf{U}''_{i^*}$ is defined by sampling $\mathbf{J}^*_{i^*} \leftarrow D_{\mathbb{Z}^{m\times m},\rho}$, and setting $\mathbf{U}''_{i^*} = \mathbf{A}' \cdot \mathbf{J}^*_{i^*}$. Then it is easy to verify $T(D_3) = \mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid}9,i^*}$.

Now, by the property of statistical distance SD, we have:

$$\begin{aligned}\mathsf{SD}(\mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid}9,i^*}, \mathcal{A}^{\mathsf{View}'}_{\mathsf{Hybrid}9,i^*-1}) &= \mathsf{SD}(T(D_3), T(D_2)) \le \mathsf{SD}(D_2, D_3) \\ &\le \mathsf{SD}(D_1, D_2) + \mathsf{SD}(D_1, D_3).\end{aligned}$$

Next we claim $\mathsf{SD}(D_1, D_3) \le \mathsf{negl}(\lambda)$. To see this, we observe that $\mathbf{U}''_{i^*} = \mathbf{A}' \cdot \mathbf{J}^*_{i^*}$, and $\mathbf{J}^*_{i^*}$ is never explicitly used by the $\mathsf{Sampler}$-1 (both D_1 and D_3). Therefore, by Lemma 2.7, the marginal distribution of $\mathbf{U}''_{i^*}$ is statistically close to random, which is exactly the distribution of $\mathbf{U}'_{i^*}$ in D_1. This thus proves the claim.

Together with Theorem 4.2 which shows $\mathsf{SD}(D_1, D_2) \le \mathsf{negl}(\lambda)$, the proof is complete.

□

Lemma B.11 Hybrid 10 *and* Hybrid 11 *are statistically indistinguishable.*

Proof. The proof follows similarly as the proof of indistinguishability between Hybrid 0 and Hybrid 1. □

Lemma B.12 Hybrid 11 *and* Hybrid 12 *are statistically indistinguishable.*

Proof. We show the indistinguishability between the two hybrids in an information theoretical way. The only difference between the two views $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid}11}$ and $\mathcal{A}^{\mathsf{View}}_{\mathsf{Hybrid}12}$ is the way to generate the message encoding vectors $\boldsymbol{b}_k$ in the cipertext elements $\boldsymbol{\beta}_{1,k}$ for $k \in [N]$. Formally, the elements $\{\widetilde{\boldsymbol{\beta}}_k\}_{k\in[N]}$ are truly random in both the two hybrids now, $\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}}$ for 1-key query $\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}$ is generated under the constrain $\begin{bmatrix} \mathbf{A} & \mathbf{A}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}} \\ \boldsymbol{z}^\top & \boldsymbol{u}^\top_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}} \end{bmatrix} \cdot (\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}}) = \begin{bmatrix} \sum_{k\in\Delta^*_{\hat{i}}} \mathbf{P}_k \\ \sum_{k\in\Delta^*_{\hat{i}}} \widetilde{\boldsymbol{\beta}}^\top_k \end{bmatrix}$, and the condition for correctness of decryption is $(\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}})^\top \cdot \begin{pmatrix} \boldsymbol{z}^\top \\ \boldsymbol{u}^\top_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}} \end{pmatrix} = \sum_{k\in\Delta^*_{\hat{i}}} \widetilde{\boldsymbol{\beta}}^\top_k$. Then $\widetilde{\boldsymbol{\beta}}_k$ acts as one-time pad to hide $\boldsymbol{b}_k$ in $\boldsymbol{\beta}_{i,k}$ if there is no 1-key query been asked, and $\mathcal{A}$ only learns the value $\sum_{k\in\Delta^*_{\hat{i}}} b_k = \lceil q/2 \rceil \mu$ if $\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}}$ if the secret key of the query $\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}$ is provided.

As for any $\hat{i} \in Q$, the summation of $\sum_{k\in\Delta^*_{\hat{i}}} b_k$ is set to be equal in these two hybrids. Therefore, the two views are statistically indistinguishable. □

Lemma B.13 Hybrid 12 *and* Hybrid 13 *are statistically indistinguishable.*

Proof. We show the view of adversary $\mathcal{A}$ in Hybrid 12 is statistically indistinguishable from the view in Hybrid 13, the ideal experiment.

For $\mathcal{A}^{\mathsf{View}} = \big(\mathsf{QPH.mpk}, \{\mathsf{sk}_{\widehat{C}_i \circ \mathsf{IP}_{\gamma_i}}\}_{i\in\mathsf{poly}}, \{\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}}\}_{\hat{i}\in[Q]}, \mathsf{ct}^*\big)$, we first observe that the distributions of $\mathsf{QPH.mpk}$, and $\{\mathsf{sk}_{\widehat{C}_i \circ \mathsf{IP}_{\gamma_i}}\}_{i\in\mathsf{poly}}$ and the elements $\boldsymbol{\beta}_0, \{\boldsymbol{u}_i\}, \{\boldsymbol{v}_j\}$ of ct^* in these two hybrids are identical. It remains to show the distributions of $\{\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}}\}_{\hat{i}\in[Q]}$, and $\{\boldsymbol{\beta}_{1,k}\}$ in ct^* in the two hybrids are statistically indistinguishable.

To see this, we first observe that the distributions of ct^* in the two hybrids are identical by a similar argument as Lemma B.12. Next, we show the distributions of $\{\mathsf{sk}_{\widehat{C}^*_{\hat{i}} \circ \mathsf{IP}_{\gamma_{\hat{i}}}} = (\Delta^*_{\hat{i}}, \mathbf{K}_{\hat{i}})\}_{\hat{i}\in[Q]}$ in the two hybrids are statistically indistinguishable. We first consider the case where there is no pre-challenge 1-key query. In Hybrid 12, $\{\Delta^*_i\}_{i\in[Q]}$ are prepared in $\mathsf{QPH.Setup}$ phase, and $\{\Delta'_i\}_{i\in[Q]}$ are chosen in the encryption algorithm. In Hybrid 13, $\{\Delta_i, \Delta'_i\}_{i\in[Q]}$ are chosen in $\mathsf{QPH.Enc}$ (Sim_3) phase. The distributions of $\{\Delta^*_i\}_{i\in[Q]}$ in the two hybrids are identical, since there is no secret key for 1-key query specified before the challenge.

Similarly, we consider the case that $\mathcal{A}$ had made Q' 1-key pre-challenge queries. In Hybrid 12, $\{\Delta^*_i\}_{i\in[Q]}$ are prepared in $\mathsf{QPH.Setup}$ phase. In Hybrid 13, the random subsets $\{\Delta_i\}_{i\in[Q']}$ corresponding to the pre-challenge 1-key queries are specified in Sim_2 phase. Thus, $\{\Delta_i\}_{i\in[Q']}$ in the two hybrids are identical, as these subsets are specified before the challenge. In Hybrid 13, $\{\Delta_i\}_{i\in[Q'+1,Q]}$ corresponding to the post-challenge 1-key queries are specified in Sim_3 phase. By a simple observation, the distributions of $\{\Delta_i\}_{i\in[Q'+1,Q]}$ in the two hybrids are identical.

On the other hand, the distributions of $\mathbf{K}_i$ in the two hybrids are statistically close, since they are both generated by the two-stage sampling method defined in Section 4. This completes the proof of Lemma B.13. □

This completes the security proof. □

C Supplementary Material for SA-Secure PHPE in Section 5.3

In this section, we first argue the correctness in C.1. Then, we present the detailed security proof in Section C.2.

C.1 Correctness of SA-Secure PHPE in Section 5.3

It is not hard to verify the correctness of the above transformation, so we just describe the most critical design idea. We first observe that the decryption keys of the PKE allow the $\mathsf{PH}_{\mathsf{SA}}.\mathsf{Dec}$ to learn $\{L_{i,r_i}\}_{i\in[t]}$ and $\{L'_{i,r'_i}\}_{i\in[\ell]}$ from $\{\mathsf{ct}_{i,b}\}_{i\in[t],b\in\{0,1\}}$ and $\{\mathsf{ct}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ only. This information together with ct_1, forms a ciphertext with the shifted private/public attributes of the underlying $\mathsf{PH}_{\mathsf{Sel}}$, i.e., $\mathsf{PH}_{\mathsf{Sel}}.\mathsf{Enc}((\boldsymbol{x}\oplus\boldsymbol{r}, \boldsymbol{y}\oplus\boldsymbol{r}'), \mu)$. Since the key generation algorithm also queries the underlying $\mathsf{PH}_{\mathsf{Sel}}.\mathsf{KeyGen}$ with a shifted function, i.e.,

$f_{\boldsymbol{r},\boldsymbol{r}'}(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{x}\oplus\boldsymbol{r},\boldsymbol{y}\oplus\boldsymbol{r}')$, we have $f_{\boldsymbol{r},\boldsymbol{r}'}(\boldsymbol{x}\oplus\boldsymbol{r},\boldsymbol{y}\oplus\boldsymbol{r}') = f(\boldsymbol{x},\boldsymbol{y})$, implying the correctness of the overall decryption.

C.2 Security of SA-Secure PHPE in Section 5.3

We present the following theorem to summarize the security the transformation achieves. As the proof is similar in spirit with the prior work [21], we will just present a sketch, which suffices to demonstrate the core ideas.

Theorem (Restatement of Theorem 5.6) *Assume that* PKE *is semantically secure, and* $\mathsf{PH}_{\mathsf{Sel}}$ *is* (q_1,q_2)-Sel-SIM *secure for private-public attribute space* $\{0,1\}^t \times \{0,1\}^\ell$*, message space* $\mathcal{M}$*, and function class* $\mathcal{F}$ *that is closed under bit-shift on* $\{0,1\}^t \times \{0,1\}^\ell$*. Then the scheme* $\mathsf{PH}_{\mathsf{SA}}$ *is* (q_1,q_2)-SA-SIM *secure for the same attribute and message spaces and the function class* $\mathcal{F}$*.*

Proof (sketch). First, we show how to construct the simulator $\mathsf{Sim}' = (\mathsf{Sim}'_1, \mathsf{Sim}'_2, \mathsf{Sim}'_3, \mathsf{Sim}'_4)$ for $\mathsf{PH}_{\mathsf{SA}}$ from the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3, \mathsf{Sim}_4)$ for the underlying $\mathsf{PH}_{\mathsf{Sel}}$, in a black box way.

- For Sim'_1, we conduct the following steps:
  1. Choose a random vector $(\tilde{\boldsymbol{r}},\boldsymbol{r}') \leftarrow \{0,1\}^t \times \{0,1\}^\ell$, and run Sim_1 with $\boldsymbol{r}'$ as input.
  2. After getting from Sim_1 the master public key $\mathsf{mpk}_{\mathsf{Sel}}$, run $\mathsf{PKE.KeyGen}$ for $(2t+2\ell)$ times to obtain the corresponding pairs of public/secret keys, $\{\mathsf{PKE.pk}_{i,b}, \mathsf{PKE.sk}_{i,b}\}_{i\in[t],b\in\{0,1\}}$ and $\{\mathsf{PKE.pk}'_{i,b}, \mathsf{PKE.sk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$.
  3. Output $\mathsf{mpk}_{\mathsf{SA}} = (\mathsf{mpk}_{\mathsf{Sel}}, \{\mathsf{PKE.pk}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{PKE.pk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$.
- For Sim'_2, after receiving key query $f\in\mathcal{F}$ and the challenge public attribute $\boldsymbol{y}^*$, we conduct the following steps:
  1. Let $\tilde{\boldsymbol{r}}' = \boldsymbol{r}'\oplus\boldsymbol{y}^*$. Define a related function $f_{\tilde{\boldsymbol{r}},\tilde{\boldsymbol{r}}'}(\boldsymbol{x},\boldsymbol{y}) := f(\boldsymbol{x}\oplus\tilde{\boldsymbol{r}},\boldsymbol{y}\oplus\tilde{\boldsymbol{r}}')$, and then run Sim_2 with $f_{\tilde{\boldsymbol{r}},\tilde{\boldsymbol{r}}'}(\cdot,\cdot)$ and $\boldsymbol{y}^*$ as input.[18]
  2. After getting $\mathsf{sk}_{\mathsf{Sel},f}$ from Sim_2, set and output $\mathsf{sk}_{\mathsf{AD},f} = (\tilde{\boldsymbol{r}},\tilde{\boldsymbol{r}}', \{\mathsf{PKE.sk}_{i,\tilde{r}_i}\}_{i\in[t]}, \{\mathsf{PKE.sk}'_{i,\tilde{r}'_i}\}_{i\in[\ell]}, \mathsf{sk}_{\mathsf{Sel},f})$.
- For Sim'_3, after receiving (b,st) for challenge ciphertext query, we conduct the following steps:
  1. Replace the queried functions $f\in\mathsf{st}$ with $f_{\tilde{\boldsymbol{r}},\tilde{\boldsymbol{r}}'}$, and generate new state information st'.
  2. Run Sim_3 with (b,st') and $\boldsymbol{y}^*$ as input.
  3. Get the challenge ciphertext $\mathsf{ct}_{\mathsf{Sel}} = (\mathsf{ct}^{(1)}_{\mathsf{Sel}}, \mathsf{ct}^{(2)}_{\mathsf{Sel}}, \mathsf{ct}^{(3)}_{\mathsf{Sel}})$, where $\mathsf{ct}^{(1)}_{\mathsf{Sel}} = \mathsf{Enc}_1(\mu;R)$, $\mathsf{ct}^{(2)}_{\mathsf{Sel}} = \{\mathsf{ct}^{(2),i}_{\mathsf{Sel}}\}_{i\in[t]} = \{\mathsf{Enc}_2(x_i;R)\}_{i\in[t]}$, $\mathsf{ct}^{(3)}_{\mathsf{Sel}} = \{\mathsf{ct}^{(3),i}_{\mathsf{Sel}}\}_{i\in[\ell]} = \{\mathsf{Enc}_3(y_i;R)\}_{i\in[\ell]}$.
  4. Set $\mathsf{ct}_1 = \mathsf{ct}^{(1)}_{\mathsf{Sel}}$, $L_{i,\tilde{r}_i} = \mathsf{ct}^{(2),i}_{\mathsf{Sel}}$ for $i\in[t]$, and $L'_{i,\tilde{r}'_i} = \mathsf{ct}^{(3),i}_{\mathsf{Sel}}$ for $i\in[\ell]$.

[18] Here, when we denote $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ as the challenge attribute of $\mathsf{PH}_{\mathsf{SA}}$, the related challenge attribute of $\mathsf{PH}_{\mathsf{Sel}}$ is $(\tilde{\boldsymbol{r}}\oplus\boldsymbol{x}^*,\boldsymbol{r}')$. Clearly, we have $f_{\tilde{\boldsymbol{r}},\tilde{\boldsymbol{r}}'}(\tilde{\boldsymbol{r}}\oplus\boldsymbol{x}^*,\boldsymbol{r}') := f(\boldsymbol{x}^*,\boldsymbol{y}^*)$, which means 1-key and 0-key queries of $\mathsf{PH}_{\mathsf{SA}}$ and $\mathsf{PH}_{\mathsf{Sel}}$ are consistent.

5. Set $\{L_{i,1-\tilde{r}_i}\}_{i\in[t]}$ and $\{L'_{i,1-\tilde{r}'_i}\}_{i\in[\ell]}$ to be random strings with the same length as $\mathsf{ct}_{\mathsf{Sel}}^{(2),i}$ and $\mathsf{ct}_{\mathsf{Sel}}^{(3),i}$, respectively.
6. Generate $\{\mathsf{ct}_{i,b}\}_{i\in[t],b\in\{0,1\}}$ and $\{\mathsf{ct}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ just as the real encryption algorithm, i.e., encrypting $L_{i,b}$ and $L'_{i,b}$ under $\mathsf{pk}_{i,b}$ and $\mathsf{pk}'_{i,b}$, respectively.
7. Output $\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{ct}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$.

– For Sim'_4, we construct it from Sim_4 in a black box way, which is similar as the black-box construction of Sim'_2.

Next, we sketch to show the indistinguishability of real $\mathsf{PH}_{\mathsf{SA}}$ experiment and the above simulator SIM', through using a hybrid argument. In particular, we first treat the real experiment as Hybrid_0. Then in Hybrid_1, we replace $\{\mathsf{ct}_{i,1-\tilde{r}_i}\}_{i\in[t]}$ and $\{\mathsf{ct}'_{i,1-\tilde{r}'_i}\}_{i\in[\ell]}$ with the encryptions of random values. Finally, we treat the above SIM' as Hybrid_2. Clearly, the indistinguishability of Hybrid_0 and Hybrid_1 follows from the security of the used PKE scheme. And the indistinguishability of Hybrid_1 and Hybrid_2 follows naturally from that of the underlying $\mathsf{PH}_{\mathsf{Sel}}$'s real experiment and SIM. The proof can be formalized in a similar way as the prior work [21]. To avoid repeating the existing technique, here we only present this sketch and refer readers to the work [21] for the detailed steps. □

D Supplementary Material of Section 7

D.1 ALS in [5]

In this section, we recall the original ALS in [5], which is necessary for the reduction proof in Section 7.1.

– $\mathsf{ALS.Setup}(1^\lambda, 1^\ell, \mathsf{IP}, \mathcal{M})$: Given as input the security parameter λ, the class IP of inner product function indexed by vectors in $\{0,\ldots,v-1\}^\ell$, and the message space $\mathcal{M} = \{0,\ldots,p-1\}^\ell$ with $\ell, v, p \in \mathbb{N}$, do the following:
1. Choose modulus q, parameters n, m, k, ρ, σ as described in next paragraph for parameters.
2. Sample random matrices $\mathbf{A}_{\mathsf{ALS}} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$ and $\mathbf{Z}_{\mathsf{ALS}} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell\times m},\rho}$;
3. Compute $\mathbf{D}_{\mathsf{ALS}} = \mathbf{Z}_{\mathsf{ALS}} \cdot \mathbf{A}_{\mathsf{ALS}} \in \mathbb{Z}_q^{\ell\times n}$;
4. Output $\mathsf{mpk} := (\mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$ and $\mathsf{msk} := \mathbf{Z}_{\mathsf{ALS}}$.

– $\mathsf{ALS.KeyGen}(\mathsf{msk}, \boldsymbol{y})$: On input a vector $\boldsymbol{y} \in \{0,\ldots,v-1\}^\ell$, the algorithm computes and outputs $\mathsf{sk}_{\boldsymbol{y}} := \mathbf{Z}_{\mathsf{ALS}}^\top \cdot \boldsymbol{y} (\mathrm{mod} q)$.

– $\mathsf{ALS.Enc}(\mathsf{mpk}, \boldsymbol{x})$: On input a message $\boldsymbol{x} \in \{0,\ldots,p-1\}^\ell$, the algorithm conducts the following steps:
1. Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n, \boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ and $\boldsymbol{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^\ell,\sigma}$;
2. Compute $\mathsf{ct}_1 = \mathbf{A}_{\mathsf{ALS}}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_0 \in \mathbb{Z}_q^m, \quad \mathsf{ct}_{2,i} = \mathbf{D}_{\mathsf{ALS}} \cdot \boldsymbol{s} + \boldsymbol{e}_1 + \lfloor \frac{q}{k} \rfloor \cdot \boldsymbol{y} \in \mathbb{Z}_q^\ell$ for $i \in [N]$;
3. Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.

– $\mathsf{ALS.Dec}(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, \boldsymbol{y}, \mathsf{ct})$: On input $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and a secret key $\mathsf{sk}_{\boldsymbol{y}}$ for $\boldsymbol{y} \in \{0,\ldots,v-1\}^\ell$, the algorithm computes $\mu' = \langle \boldsymbol{y}, \mathsf{ct}_2 \rangle - \langle \mathsf{sk}_{\boldsymbol{y}}, \mathsf{ct}_1 \rangle (\bmod q)$, and then outputs the value $\mu \in \{0,\ldots,p-1\}$ that minimizes $|\lfloor \frac{q}{k} \rfloor \cdot \mu - \mu'|$.

Parameters setting.

- To ensure correctness of decryption, we require $q \geq 2k\ell\sqrt{\ell}v\omega(\log^2 n)$, $\rho, \sigma > \omega(\sqrt{\log m})$.
- To ensure the correctness of ReRand algorithm, we require $\sigma \geq 2\alpha q\Omega(\rho(\sqrt{\ell}+\sqrt{m}+\sqrt{n}))$ (c.f. Lemma 2.10).
- To ensure large enough entropy required in the security proof, we require $\rho > \omega(\sqrt{\log m})$, $m \geq (2n\log q + 2n)/\log(4/3)$.
- To ensure the hardness of the underlying $\mathsf{LWE}_{n,q,\alpha}$, we require $q\alpha \geq \Omega(\sqrt{n})$.

Under the parameters above, the ALS scheme described above is adaptively secure for chosen message attacks.

Theorem D.1 *Under the* $\mathsf{LWE}_{n,q,\alpha}$ *assumption, the above* ALS *is* $\mathsf{AD\text{-}IND}$*secure within the above parameters setting.*

D.2 *N*-ALS in [47]

In this section, we recall the modified N-ALS in [47], and its IND-based security notion for chosen message distributions, which is necessary for the reduction proof in Section 7.2. Finally, we present the related parameter setting and formal theorem.

- N-$\mathsf{ALS.Setup}(1^\lambda, 1^\ell, p, N)$: The algorithm conducts the following steps:
  1. Set parameters $n, m, \sigma, \sigma', q = p^e$ for some integer e;
  2. Sample random matrices $\mathbf{A}_{N\text{-}\mathsf{ALS}} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$ and $\mathbf{Z}_{N\text{-}\mathsf{ALS},i} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times m}$, for $i = 1, \ldots, N$;
  3. Compute $\mathbf{D}_{N\text{-}\mathsf{ALS},i} = \mathbf{Z}_{N\text{-}\mathsf{ALS},i} \cdot \mathbf{A}_{N\text{-}\mathsf{ALS}} \in \mathbb{Z}_q^{\ell\times n}$;
  4. Output $\mathsf{mpk} := (\mathbf{A}_{N\text{-}\mathsf{ALS}}, \{\mathbf{D}_{N\text{-}\mathsf{ALS},i}\}_{i\in[N]})$ and $\mathsf{msk} := (\{\mathbf{Z}_{N\text{-}\mathsf{ALS},i}\}_{i\in[N]})$.
- N-$\mathsf{ALS.KeyGen}(\mathsf{msk}, \boldsymbol{y})$: On input a vector $\boldsymbol{y} = (\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_N^\top)^\top \in \mathbb{Z}_p^{N\cdot\ell}$, the algorithm computes and outputs $\mathsf{sk}_{\boldsymbol{y}} := \sum_{i=1}^N \left(\mathbf{Z}_{N\text{-}\mathsf{ALS},i}^\top \cdot \boldsymbol{y}_i\right)$.[19]
- N-$\mathsf{ALS.Enc}(\mathsf{mpk}, \boldsymbol{x})$: On input a message $\boldsymbol{x} = (\boldsymbol{x}_1^\top, ..., \boldsymbol{x}_N^\top)^\top \in \mathbb{Z}_p^{N\cdot\ell}$, the algorithm conducts the following steps:
  1. Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n, \boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ and $\{\boldsymbol{e}_i\}_{i\in[N]} \leftarrow \mathcal{D}_{\mathbb{Z}^\ell,\sigma'}$;
  2. Compute $\mathsf{ct}_1 = \mathbf{A}_{N\text{-}\mathsf{ALS}}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_0 \in \mathbb{Z}_q^m$, $\quad \mathsf{ct}_{2,i} = \mathbf{D}_{N\text{-}\mathsf{ALS},i} \cdot \boldsymbol{s} + \boldsymbol{e}_i + p^{e-1} \cdot \boldsymbol{y}_i \in \mathbb{Z}_q^\ell$ for $i \in [N]$;
  3. Output $\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{2,i}\}_{i\in[N]})$.
- N-$\mathsf{ALS.Dec}(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, \boldsymbol{y}, \mathsf{ct})$: On input $\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{2,i}\}_{i\in[N]})$ and a secret key $\mathsf{sk}_{\boldsymbol{y}}$ for $\boldsymbol{y} = (\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_N^\top)^\top \in \mathbb{Z}_p^{N\cdot\ell}$, the algorithm computes $\mu' = \sum_{i=1}^N (\langle \boldsymbol{y}_i, \mathsf{ct}_{2,i}\rangle) - \langle \mathsf{sk}_{\boldsymbol{y}}, \mathsf{ct}_1\rangle (\mathrm{mod} q)$, and then outputs the value $\mu \in \mathbb{Z}_p$ that minimizes $|p^{e-1} \cdot \mu - \mu'|$.

Next, we present the IND-based security notion for chosen message distributions. For ease of exposition, we first introduce the definitions of the query mappings and *admissible mappings*.

[19] We require that all the queried $\boldsymbol{x}$ should be linearly independent modulo from each other over $\mathbb{Z}_p^{N\cdot\ell}$.

Definition D.2 *Let $t = t(\lambda)$ be a integers and $\mathcal{M}$ be the message space. $\{x_i\}_{i\in[t]} \in \mathcal{M}^t$ is a set of messages, and $f : \mathcal{M} \to \mathcal{K}$ be a function. We define the functions $(i, f) : \mathcal{M}^t \to \mathcal{K}$ as $(i, f)(x_1, ..., x_t) = f(x_i)$, and function $(i, I) : \mathcal{M}^t \to \mathcal{M}$ as $(i, I)(x_1, ..., x_t) = x_i$.*

Definition D.3 (Admissible mappings) *Let $t = t(\lambda)$ be an integer, $\mathcal{M}$ be the message space, and $\mathcal{M}_0, \mathcal{M}_1$ be two distributions over space $\mathcal{M}^t$. Let subsets $T_1, T_2 \subsetneq [t]$ such that $T_2 \cap T_1 = \emptyset$ and $|T_2 \cup T_1| < t$, and let $\{k_i\}_{i\in T_2}$ be a set of integers. We say that mappings $\{(i, I)\}_{i\in T_1}$ and $\{(i, f_{ij})\}_{i\in T_2, j\in[k_i]}$ are admissible if it holds that*

$$\begin{aligned}&\{\{(i, I)(\mathcal{M}_0)\}_{i\in T_1}, \{(i, f_{ij})(\mathcal{M}_0)\}_{i\in T_2, j\in[k_i]}\}\\ =&\{\{(i, I)(\mathcal{M}_1)\}_{i\in T_1}, \{(i, f_{ij})(\mathcal{M}_1)\}_{i\in T_2, j\in[k_i]}\}\end{aligned}$$

Remark D.4 The requirement of *admissible mappings* is that the above two distributions are identical. It can also be relaxed by requiring the two distributions are statistically or computationally close.

Next, we present the adaptive security of functional encryption for chosen message distributions through an experiment $\mathbf{Exp}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1^t)$ between an adversary and challenger:

1. **Setup**: For $i \in [t]$, challenger first computes $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{Setup}(1^\lambda)$, then sends $\{\mathsf{mpk}_i\}_{i\in[t]}$ to adversary $\mathcal{A}$.
2. **Query Phase I**: Proceeding adaptively, adversary can make any polynomial number of queries to the oracle $\mathcal{O}(\{\mathsf{msk}_i\}_{i\in[t]}, \cdot)$ of the following two kinds:
   - Function queries (i, f_{ij}): Challenger sends back $\mathsf{sk}_{f_{ij}} \leftarrow \mathsf{KeyGen}(\mathsf{sk}_i, f_{ij})$.
   - Opening queries (i, I): Challenger sends back msk_i.
3. **Challenge Phase**: Adversary $\mathcal{A}$ sends two message distributions $\mathcal{M}_0$ and $\mathcal{M}_1$ over message space $\mathcal{M}^t$ with the restriction that any queries made in **Query Phase I** are *admissible* with respect to $(\mathcal{M}_0, \mathcal{M}_1)$ (c.f. Definition D.3). The challenger chooses a random bit $b \in \{0, 1\}$, and sends ciphertext $\{\mathsf{ct}_i = \mathsf{Enc}(\mathsf{mpk}_i, x_i)\}_{i\in[t]}$ back to adversary, where $\{x_i\}_{i\in[t]} \leftarrow \mathcal{M}_b$.
4. **Query Phase II**: Adversary $\mathcal{A}$ can continue making queries as specified in **Query Phase I** as long as the queries are admissible.
5. **Guess**: Adversary $\mathcal{A}$ outputs his guess b'.

We define the advantage of adversary $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1^t)$ as

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda, 1^t) = |\mathsf{Pr}[\mathbf{Exp}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1^t) = 1] - 1/2|$$

Definition D.5 *We say a functional encryption scheme Π is adaptively secure for chosen message distributions security if for any polynomial $t = t(\lambda)$, and any* PPT *adversary $\mathcal{A}$, we have $\mathbf{Adv}_{\mathcal{A}}(1^\lambda, 1^t) \leq \mathsf{negl}(\lambda)$.*

Finally, we present the parameter setting and security theorem of the above N-ALS.

Parameters setting.

- To ensure correctness of decryption, we require $p^{e-1} > 2Np^2m\ell\alpha q\sqrt{n}$.
- To ensure the correctness of ReRand algorithm, we require $\sigma^* \geq pm$ (c.f. Lemma 2.10).
- By the property of ReRand algorithm, we have $\sigma^{'} = 2\sigma^*\sigma$ (c.f. Lemma 2.10).
- To ensure small enough reduction loss for the ReRand algorithm, we require $\alpha q > \sqrt{\lambda + tN^2\ell^2 \log p}$ (c.f. Lemma 2.10).
- To ensure large enough entropy required in the security proof, we require $m \geq 2N\ell + eN\ell(n+1) + 3\lambda$.

Under the parameters above, the N-ALS described above is adaptively secure for chosen message distributions.

Theorem D.6 *Under the* LWE *assumption, the above* N-ALS *is adaptively secure for chosen message distributions, assuming all the secret key queries* $\boldsymbol{y}$ *to* N-ALS.KeyGen *are linearly independent.*