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], and Zhedong Wang[3]

[1] School of Computer Science, Shaanxi Normal University, Xi'an, China
[2] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China
laiqq@snnu.edu.cn
[3] Florida Atlantic University, Boca Raton, FL, USA
{fenghao.liu,wangz}@fau.edu

Abstract. This work proposes a new 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) [13,35] 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., [2,4–6,21,26,27] 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 [1,2,13,27]. Particularly, each message u consists of two parts, namely $u := (\mathsf{x}, \mu)$, where x is some index (or attribute)[1],

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

A. Canteaut and F.-X. Standaert (Eds.): EUROCRYPT 2021, LNCS 12696, pp. 498–527, 2021.
https://doi.org/10.1007/978-3-030-77870-5_18

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 [13], 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 the full version of this paper.

To capture security, there have been notions of indistinguishable-based (IND) and simulation-based (SIM) definitions proposed and studied in the literature since [13]. As raised by [13], 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 [13] 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 the class of general functions, the ciphertext size must grow linearly with the number of pre-challenge key queries. Therefore it is impossible to achieve the notion $(\mathsf{poly}, \mathsf{poly})$-SIM security (allowing an unbounded number of both 1 and 0-keys) for general functions.

Despite these lower bounds, the work [27] identified important feasible settings for SIM-based security, by proposing new constructions in the setting of single challenge ciphertext and bounded collusion. More specifically, [27] 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 of the function), and their size grows with a multiplicative factor of $O(Q^4)$. Even though a recent work [10] improved the multiplicative factor to $O(Q)$, their ciphertexts are still not succinct, prohibiting other important applications, such as reusable garbled circuits [26]. Thus, improvements in this dimension would be very significant.

A subsequent work [26] 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 [26] has drawbacks in the following two aspects. First, the single-key FE of [26] 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 [26] can be bootstrapped to Q key FE using the compiler of [27], 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 induces 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[2] subclass of key queries (i.e., 0-keys), where SIM-based security is feasible beyond bounded collusion. Recently, the work [10] 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 [27] 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 challenge ciphertext; (3) no more 1-key is allowed for post-challenge phase. How to bridge the gap between the two methods [2,27] 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 satisfac-

[2] 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.

tory) by the generic complexity leveraging argument as also pointed out by [26]. 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[3], 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 [13]. 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 [11] 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.

1.1 Our Contributions

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

[3] 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.

Contribution 1. First we propose a new two-stage lattice two-stage sampling technique, generalizing the prior GPV type two-stage sampling [24]. 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 sampling technique, 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][4] 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 the 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 [17,30], 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 transformation of [2, 29], 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 [26]. 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 by [10]. 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 transformation of [2,29].
  Importantly, our transformation inherits many nice properties in [2], e.g., the succinctness of the ciphertexts is preserved. Thus, our resulting FE has

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

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 [10], which incurs a *multiplicative* $O(Q)$ blowup in the ciphertexts.

Table 1. Comparison of prior work of single key SIM-secure public-key FE.

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

($*$) 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.

Table 2. Comparison of other private index SIM-secure public-key FE.

Ref.	(1-key, 0-key)	(Pre, Post)-Challenge	Index	Succinct ct	Ciphertext size
[27]	(Q, Q)	(✓, ✓)	AD	✗	$\times\ O(Q^4)$
[26]+ [27]	(Q, Q)	(✓, ✗)	Sel†	✓	$\times\ O(Q^4)$
[6]	(Q, Q)	(✓, ✗)	AD	✓ for NC1	$+\ O(Q^2)$
[2]	(Q, poly)	(✗, ✗)*	SA†	✓	$+\ O(Q^2)$
[2]+ [10]‡	(Q, poly)	(✗, ✗)*	SA†	✓	$\times\ O(Q)$
Ours	(Q, poly)	(✓, ✓)	SA†	✓	$+\ O(Q)$

($*$) 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.
(†) Similar to Table 1.
(‡) The generic method in [10] 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 IB × IP and AB × 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].

Table 3. Comparison of public index IND-based construction.

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

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 be useful in 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 [1–3,11,14,24,28,29], 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 by using the $\mathbf{G}$-trapdoor technique [33]. Particularly, if we have the $\mathbf{G}$ matrix [33] in the right, 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 [3,33]. However, our task (and the security proofs in this work) does not have $\mathbf{G}$ 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 [24] 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 prior 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 $[\mathbf{A}|\mathbf{AR}]\boldsymbol{y} = \boldsymbol{u}$. 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 preserves succinctness of 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 from [29] 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 (3) does not support post-challenge 1-key queries.
- Fourth, Step (3) incurs an additive overhead of $O(Q^2)$, which is incomparable with the multiplicative $O(Q)$ overhead the recent work by [10].

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 [11]. 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 $[\mathbf{A}|\mathbf{B}_{C_f}] = [\mathbf{A}|\mathbf{A}\mathbf{R}_{C_f} + C_f(\mathbf{x}^*)\mathbf{G}]$. 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 [17,30], the FE can be upgraded to semi-adaptive security[5]. 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 $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell$,

[5] The reason why [2] cannot apply the light-weight method is because its basic construction only achieves very selective security, whereas the technique of [17,30] can be applied to the selective security only over index.

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 [27], 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 [10]. We identify that this more efficient cover-freeness suffices for the rest of the proof.

Finally, we show how to handle 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 share the plaintext $\mu \in \{0, 1\}$, more precisely, $\frac{q}{2}\mu$, into $\mu_1, \ldots, \mu_N$, such that any subset Δ with some fixed cardinality would recover the message, i.e., $\frac{q}{2}\mu = \sum_{i\in\Delta} \mu_i$. 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 proof insight, we show that given all secret keys of the form $(\Delta, \mathbf{Y})$, one can only learn $\sum_{i\in\Delta} \mu_i = \frac{q}{2}\mu$ but nothing more. By using this fact, we can design a 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 $\sum_{\Delta_i} \mu_i = q/2$, and $\sum_{\Delta'_i} \mu_i = 0$. Thus in the post-challenge stage, the 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) as [2].

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 IB × IP and semi-adaptive AB × IP. We further elaborate on this setting in Sect. 6. 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 ($i\mathcal{O}$) [21], achieving the notion of $(\mathsf{poly}, \mathsf{poly})$-IND adaptive security via [9]. Even though

recently there has been substantial progress for instantiating $i\mathcal{O}$ [15,22,31], 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 [13,27], 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 $i\mathcal{O}$ [15,22,31].

In [18], Canetti and Chen show that a single key SIM-secure private-key FE suffices to construct reusable garbled circuits. Compared with the reusable garbled circuits derived from our (Q, poly)-SA-SIM FE with $Q = 1$,[6], the construction in [18] achieves the stronger adaptive security with respect to index without the complexity leveraging argument, yet can only support either a pre- or post-challenge key query for a NC1 circuit, rather than a general circuit.

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 implicit input for all algorithms 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 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 denote $|u|$ 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 denote their statistical distance as $\mathsf{SD}(X,Y) = \frac{1}{2}\sum_{d\in D} |X(d) - Y(d)|$, 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 $\tilde{\mathbf{R}}$ to denote its Gram-Schmidt orthogonalization. In addition,

[6] Notice that the reusable garbled circuits following from our SIM-secure FE can achieve SA-SIM security, and support general circuits and any arbitrary pre- and post-challenge key query, for one query.

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

Lemma 2.1 ([20]). *Let $\mathbf{X} \in \mathbb{R}^{n\times m}$ be a subgaussian random matrix with parameter s. There exists a universal constant $c \approx 1/\sqrt{2\pi}$ such that for any $t > 0$, we have $s_1(\mathbf{X}) \leq c \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $\frac{2}{e^{\pi t^2}}$.*

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

Lemma 2.2 (Cover-Freeness [27]). *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 $\mathsf{Pr}\left[\Delta_i \backslash \left(\bigcup_{j\neq i} \Delta_j\right) \neq \phi\right] = 1 - 2^{-\Omega(\kappa)}$, where the probability is over the random choice of subsets $\Delta_1, \cdots, \Delta_Q$.*

2.2 Gaussians on Lattices

Due to space limit, we defer well-known background on lattices to the full version of this paper. Here we just give some useful preliminaries of 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 [34], 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 $\rho_{1/s}(\Lambda^* \backslash \{0\}) \leq \epsilon$.*

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

Lemma 2.4 ([24]). *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 ([24] and Full Version of [32]). *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 $\eta_\varepsilon(\Lambda_q^\perp(\mathbf{A}|\mathbf{AR})) \leq \omega(\sqrt{\log m})$.

Due to space limit, we defer the proof of Lemma 2.6 to full version.

Lemma 2.7 ([24], 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 ([34], 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 $\Pr_{\boldsymbol{x} \leftarrow \mathcal{D}_{\Lambda, s, c}}[\|\boldsymbol{x} - \boldsymbol{c}\| > s\sqrt{m}] \leqslant \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-m}$.*

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

Learning With Errors. The Learning with Errors problem, or LWE, is the problem of determining a secret vector over $\mathbb{Z}_q$ 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 2.10 ([37]). *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 [16,36,37] show that the LWE assumption is as hard as (quantum or classical) solving GapSVP and SIVP under various parameter regimes.

2.3 Lattice Trapdoor and Gaussian Sampling

Gadget Matrix. We recall the "gadget matrix" $\mathbf{G}$ defined in [33]. 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 2.11 (Theorem 4.1, [33]). *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 2.12 (Trapdoor Generation [8,33]). *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 2.13 (SampleLeft [3,19]). *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 2.14 (SampleRight [33]). *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.*

2.4 Partially Hiding Predicate Encryption

We recall the notation of partially hiding predicate encryption (PHPE) proposed by [29], 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
(PHPE.Setup, PHPE.Enc, PHPE.KeyGen, PHPE.Dec):

- $\mathsf{PHPE.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{F}, \mathcal{M}) \rightarrow (\mathsf{PHPE.mpk}, \mathsf{PHPE.msk})$. 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.
- $\mathsf{PHPE.Enc}(\mathsf{PHPE.mpk}, (\boldsymbol{x}, \boldsymbol{y}), \mu) \rightarrow \mathsf{ct}_{\boldsymbol{y}}$. 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 $\mathsf{ct}_{\boldsymbol{y}}$.
- $\mathsf{PHPE.KeyGen}(\mathsf{PHPE.msk}, f) \rightarrow \mathsf{sk}_f$. The key generation algorithm gets as input PHPE.msk and a function $f \in \mathcal{F}$. It outputs a secret key sk_f.
- $\mathsf{PHPE.Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{\boldsymbol{y}}, \boldsymbol{y})) \rightarrow \mu \vee \perp$. 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., $f(\boldsymbol{x}, \boldsymbol{y}) = 0$, $\Pr[\mathsf{PHPE.Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{\boldsymbol{y}}, \boldsymbol{y})) \neq \perp] \leq \mathsf{negl}(\lambda)$.

Due to space limit, we defer the full security definition of PHPE to full version.

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 [2,13,27]. 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. Due to space limit, we defer the detailed security notions of interests on these two cases and comparisons between the notions in related work to the full version of this paper.

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 ([24]). *For any prime q, integers integer $n \geq 1$, $m \geq 2n \log q$, $s \geq \omega(\sqrt{\log m})$, 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.[7] 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 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.

[7] 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.

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 $\mathbf{T_A}$ of $\mathbf{A}$, using the SampleLeft algorithm as Lemma 2.13 (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 Index

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 Sect. 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 Sect. 5.2. In Sect. 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 Sect. 5.4.

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 $t \in \mathbb{N}, q \in \mathbb{N}$ and $t = t' \log q$. 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 $\mathcal{F}$. 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 [2,29], which directly defined the input $\boldsymbol{x}$ in the domain $\mathbb{Z}_q^{t'}$. In full version, we show that this formulation can also achieve the same effect as the prior work [2,29] 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 Sect. 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, 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 Sect. 4 for the key generation. Our scheme achieves $(1, \mathsf{poly})$-Sel-Sim security, whose formal definition is deferred to the full version of this paper due to the space limit, 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 $\mathbf{U} = \mathbf{P} - \mathbf{AJ} (\mathrm{mod} q)$.
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 $\boldsymbol{b} = [0, \cdots, 0, \lceil q/2 \rfloor \mu]^\top \in \mathbb{Z}_q^m$. 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 Sect. 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).*

Due to space limit, we defer the correctness, parameter setting and the detailed proof of Theorem 5.3 to the full version of this paper.

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 Sect. 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 [10]. 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 [10], 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 $\mathbf{U} = \mathbf{P}_\Delta - \mathbf{AJ}$.
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 $[\mathbf{A}|\mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\mathbf{G}] \cdot \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix} = \mathbf{U} \bmod q$.
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 Sect. 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.*

Due to space limit, we defer the correctness and parameter setting to the full version of this paper. Additionally, we just describe the simulator Sim for Theorem 5.5 here, and defer the detailed proof to the full version.

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

5.3 Semi-Adaptively Secure Partially Hiding Predicate Encryption

In this section, we show how to upgrade our PHPE in Sect. 5.2 from (Q, poly)-Sel-SIM security to (Q, poly)-SA-SIM security. Technically, we follow the idea of [17], 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 $\{\mathsf{PKE.pk}_{i,b}, \mathsf{PKE.sk}_{i,b}\}_{i\in[t],b\in\{0,1\}}$, $\{\mathsf{PKE.pk}'_{i,b}, \mathsf{PKE.sk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ from the scheme PKE.
- Sample a random string $(\boldsymbol{r}, \boldsymbol{r}') \in \{0,1\}^t \times \{0,1\}^\ell$.
- Finally output $\mathsf{mpk}_{\mathsf{SA}} = (\mathsf{mpk}_{\mathsf{Sel}}, \{\mathsf{PKE.pk}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{PKE.pk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$ as the master public key, and keep private $\mathsf{msk}_{\mathsf{SA}} = (\mathsf{msk}_{\mathsf{Sel}}, \{\mathsf{PKE.sk}_{i,b}\}_{i\in[t],b\in\{0,1\}}, \{\mathsf{PKE.sk}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \boldsymbol{r}, \boldsymbol{r}')$ 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 $(\boldsymbol{r}, \boldsymbol{r}', \{\mathsf{PKE.sk}_{i,r_i}\}_{i\in[t]}, \{\mathsf{PKE.sk}'_{i,r'_i}\}_{i\in[\ell]}, \mathsf{sk}_{\mathsf{Sel},f})$ 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 $\{\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_{i,b}, L_{i,b})\}_{i\in[t],b\in\{0,1\}}$ and $\{\mathsf{ct}'_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}'_{i,b}, L'_{i,b})\}_{i\in[\ell],b\in\{0,1\}}$.
- 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* $\mathcal{F}$*.*

Due to space limit, we defer the correctness and the proof of Theorem 5.6 to the full version of this paper.

5.4 (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. Due to space limit, we just describe the theorem from [2], and defer the detailed procedure to the full version.

Theorem 5.7. *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 Sect. 5, and* FHE *be a secure fully-homomorphic encryption scheme. Then there exists a functional encryption that is* (Q, poly)-SA-SIM *secure for the class* $\mathcal{C} \times \{I\}$*.*

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 5.8. *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* $\mathcal{C} \times \{I\}$*.*

6 Constructions of FE with Public Index

We notice that our two-stage sampling technique in Sect. 4 can be further used to derived several new feasibilities of FE with public index for the following two function classes.

- The first scheme is IB-FEIP that achieves $(1, \mathsf{poly})$-AD-IND security, i.e., a public-index FE for the class $\mathsf{IB} \times \mathsf{IP}$. Detailed definitions are deferred to the full version. This particularly improves the prior analysis of Abdalla et al. [1], who can only achieve the selectively security. As we discussed in full version of this paper, $(1, \mathsf{poly})$-AD-IND is the best we can achieve for the IB predicates as there is only one 1-key corresponding to the challenge index.
  Our construction follows the same design paradigm as [1], except we use the adaptively secure encoding of matrices by [3] and adopt our new sampling algorithm in Sect. 4 for the key generation.

– The second scheme is a generalization of the first scheme that achieves (Q, poly)-SA-IND secure AB-FEIP for any polynomially bounded Q, for general predicate classes (i.e., bounded depth boolean circuits). This new feasibility result is beyond what the prior technique of [1] can achieve.

Due to space limit, we defer the detailed constructions and security proofs of our new IB-FEIP and AB-FEIP to the full version.

Acknowledgements. We would like to thank the anonymous reviewers of Eurocrypt 2021 for their insightful advices. Qiqi Lai is supported by the National Key R&D Program of China (2017YFB0802000), the National Natural Science Foundation of China (61802241, U2001205, 61772326, 61802242), the Natural Science Basic Research Plan in Shaanxi Province of China (2019JQ-360), 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 an NSF Award CNS-1657040 and an 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. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 467–497. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_16
2. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 3–35. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_1
3. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert [25], pp. 553–572
4. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_28
5. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
6. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 173–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_7
7. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptology **9**(3), 169–203 (2015)
8. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Theory Comput. Syst. **48**(3), 535–553 (2010)
9. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R. [23], pp. 657–677

10. Ananth, P., Vaikuntanathan, V.: Optimal bounded-collusion secure functional encryption. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part I. LNCS, vol. 11891, pp. 174–198. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_8
11. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
12. Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. ACM Press, June 2013
13. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
14. Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_29
15. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: 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
16. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh et al. [12], pp. 575–584
17. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: unbounded attributes and semi-adaptive security. In: Robshaw, K. [38], pp. 363–384
18. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 446–476. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_16
19. Cash, D. Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert [25], pp. 523–552
20. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 335–352. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_19
21. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, pp. 40–49. IEEE Computer Society Press, October 2013
22. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010 (2020). https://eprint.iacr.org/2020/1010
23. Gennaro, R., Robshaw, M.J.B. (eds.): CRYPTO 2015, Part II. LNCS, vol. 9216. Springer, Heidelberg, August 2015
24. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.), 40th ACM STOC, pp. 197–206. ACM Press, May 2008
25. Gilbert, H., (ed.) EUROCRYPT 2010, volume 6110 of LNCS. Springer, Heidelberg, May/June 2010. https://doi.org/10.1007/978-3-642-13190-5
26. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh et al. [12], pp. 555–564
27. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11

28. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh et al. [12], pp. 545–554
29. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R. [23], pp. 503–523
30. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14
31. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003 (2020). https://eprint.iacr.org/2020/1003
32. Lai, Q., Liu, F.-H., Wang, Z.: Almost tight security in lattices with polynomial moduli – PRF, IBE, All-but-many LTF, and More. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 652–681. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_22
33. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
34. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS, pp. 372–381. IEEE Computer Society Press, October 2004
35. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint archive, Report 2010/556 (2010). https://eprint.iacr.org/2010/556
36. Peikert, C.: Public-key Cryptosystems from the Worst-case Shortest Vector Problem: Extended Abstract. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 333–342. ACM Press, May/June (2009)
37. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.), 37th ACM STOC, pp. 84–93. ACM Press, May 2005
38. Robshaw, M., Katz, J. (eds.): CRYPTO 2016, Part III. LNCS, vol. 9816. Springer, Heidelberg, August 2016