Generic-Group Barriers for Function-Hiding and Multi-Input Functional Encryption

Mohammad Hajiabadi [1] Roman Langrehr* [1] Mingyuan Wang [2]

[1] University of Waterloo
{mdhajiabadi,roman.langrehr}@uwaterloo.ca
[2] NYU Shanghai
mingyuan.wang@nyu.edu

Abstract

We show that private-key function-hiding inner-product functional encryption (FH-IPFE) is impossible in the generic group model (GGM). This impossibility extends to (non-compact) two-input quadratic functional encryption (QFE) under a weak security notion that allows only a single key corruption. Our results apply both to the variant where decryption outputs the result directly, and to the variant where the result is encoded in the exponent of a group element.

Our results hold in both Maurer's and Shoup's model, with different tradeoffs. In Maurer's model, we prove that FH-IPFE over $\mathbb{Z}_q^n$ cannot be realized even when q^n is polynomially bounded. Here, q denotes the modulus of the inner-product functionality, not the order of the underlying group. This stands in sharp contrast to non-function-hiding FE, which can be constructed from minimal assumptions (one-way functions in the private-key setting and public-key encryption in the public-key setting) whenever the set of functions is polynomially bounded. We extend this impossibility to Shoup's model when q^n is super-polynomial. Conceptually, our proof simulates any construction in Shoup's model as one in Maurer's model equipped with a random oracle. Our techniques may be of independent interest, offering a *general* method for upgrading other impossibility results from Maurer's model to Shoup's model.

We match these negative results with two positive ones. First, we show that *one-sided bounded* FH-IPFE (i.e., either the number of key queries or the number of encryption queries is bounded) can be realized from one-way functions. Second, when both the number of key queries and encryption queries are bounded, we show the resulting notion of FH-IPFE can be achieved information-theoretically. These positive results show that our impossibility precisely characterizes the threshold for FH-IPFE.

Keywords: Black-box impossibility, Functional encryption

1 Introduction

Functional Encryption (FE) [37, 19, 34] is an advanced encryption paradigm that enables controlled access to encrypted data. In particular, a secret key sk_f is associated with a function f so that, given a ciphertext ct encrypting a message x, the keyholder can decrypt to learn $f(x)$ but nothing

*Part of the work was done while the author was affiliated with ETH Zurich.

else. In the private-key setting, an important and desirable property is function-hiding (FH-FE) [39], which requires the secret key sk_f to reveal no information about f. Function-hiding is generally impossible in the public-key setting, as sk_f effectively grants oracle access to the function $f(\cdot)$. Such a notion enables secure delegation of function evaluation and has found real-world applications, including privacy-preserving machine learning and computation of statistics [29, 6], biometric authentication [29], DNA similarity testing [42], and keyword search [15].

Despite being a natural extension to functional encryption, function-hiding functional encryption remains elusive to construct. To the best of our knowledge, the only known constructions of FH-FE from 'standard assumptions' are for the inner-product function family. For more powerful functionalities, only constructions based on FE for general circuits are known [20] (which, in turn, requires heavy assumptions such as indistinguishability obfuscation). In FH-IPFE, both the function f and the message x are vectors in $\mathbb{Z}_q^n$, and decryption reveals only the inner product $\langle f, x\rangle \in \mathbb{Z}_q$. Even for this seemingly simple family, the only known constructions of FH-IPFE rely on pairing-based assumptions [16, 23, 44, 3, 29, 42]. This is in sharp contrast to its non-function-hiding variant, which can be built from various assumptions, including Decisional Diffie-Hellman (DDH) [2, 16, 11, 5, 14, 3, 21, 1, 42, 26, 8, 4], Learning With Errors (LWE) [11, 10] and Decision Composite Residuosity (DCR) [11].

Given this lack of progress, several recent works [45, 41] attempted to understand the difficulties in constructing FH-IPFE. However, these works have focused exclusively on the lattice setting. Consequently, the current state-of-the-art leaves the following natural question completely unexplored.

Is Function-Hiding Inner-Product Functional Encryption possible from pairing-free groups?

Dimensions to FE. FE schemes can be analyzed along several dimensions — such as bounded vs. unbounded key corruptions/collusions, polynomial vs. exponential function families, and compact vs. non-compact ciphertexts, to name a few. Bounded collusions mean the number of adversary's secret-key queries is fixed in advance. Compactness requires that the ciphertext size grows only with the plaintext length, and not with $|f|$ of the underlying function $f \in \mathcal{F}$. Polynomial vs exponential-sized function families refer to the number of all possible functions. It is known that non-compact, bounded-collusion FE for all circuits can be constructed from minimal assumptions. Regarding compactness, non-compact quadratic FE (QFE) schemes can be built from DDH via a reduction to IPFE by linearizing the QFE function. However, compact QFE constructions are only known under stronger assumptions such as pairings [14, 46, 9, 43].

A fundamental question is whether seemingly "easy" properties — such as bounded collusions, non-compactness, or polynomial-sized function families — remain trivial for non-vanilla FE, like function-hiding or multi-input FE.

1.1 Our Contributions

We show that private-key FH-IPFE is impossible in the generic group model (GGM). We prove this impossibility in both Maurer's model [33] and Shoup's model [40] with different tradeoffs. Furthermore, we establish a new connection: Any non-compact, two-input QFE scheme that is secure against only a single-key corruption implies the existence of FH-IPFE. This implication allows us to rule out such QFE schemes in the GGM as well. To the best of our knowledge, this connection

has never been observed previously. Finally, we match our negative FH-IPFE with positive results for bounded collusions in FH-IPFE.

New insights on FE. By studying FH-IPFE, we uncover subtle interactions involving the most permissive variants of required FE dimensions: bounded collusions, non-compactness, and polynomial-sized function families. Our results show that even in this highly relaxed regime, adding non-vanilla capabilities such as function-hiding or multi-input functionality leads to fundamental obstacles, even in the GGM. While it is known that for FE for general circuits some of these properties become non-trivial when combined with advanced capabilities [12, 17, 30, 25], our work demonstrates that such tensions already arise in significantly simpler settings. We elaborate below.

Impossibility in Maurer's model. We first prove that FH-IPFE is impossible in Maurer's GGM in the strongest sense.

Informal Theorem 1. There is no unbounded collusion FH-IPFE over $\mathbb{Z}_q^n$ in Maurer's GGM for any $n, q \geqslant 2$ with $(n, q) \neq (2, 2)$, achieving correctness probability $\frac{1}{2} + \frac{1}{\mathsf{poly}}$ or $\frac{2}{q} + \frac{1}{q^{n-1}} + \frac{1}{\mathsf{poly}}$ for a polynomial poly.[1]

Perhaps surprisingly, our theorem holds even if q^n is only polynomially large. Note that, when the function family is polynomial sized, one can generically build private-key FE by letting each functional key sk_f be a secret key for a standard encryption scheme, and the master secret key msk contain all such sk_f. Encrypting a message x under msk outputs all the ciphertexts $\{\mathrm{Enc}(\mathsf{sk}_f, f(x))\}_f$.

At first glance, this may suggest function-hiding is trivial for polynomial-size families, since sk_f appears independent of f, seemingly supporting function privacy. However, this intuition is flawed. A function-hiding FE scheme must satisfy semantic security for functional keys, requiring the key generation process to be randomized. The above scheme is trivially broken if an adversary can query the same function key twice. Our result shows that this limitation is not merely an artifact of this naïve construction but is, in fact, inherent, even in the GGM.

Shoup impossibility. We further extend our impossibility to Shoup's model [40].

Informal Theorem 2. For any $q \geqslant 3$, $n \geqslant 4$ where q^{-n} is negligible, no unbounded collusion FH-IPFE over $\mathbb{Z}_q^n$ exists in Shoup's GGM with correctness $\frac{3}{q} + \frac{1}{\mathsf{poly}}$ for a polynomial poly.

An (FH)-IPFE with correctness probability $\frac{1}{q}$ is trivial to build, so we must assume correctness at least $\frac{1}{q} + \frac{1}{\mathsf{poly}}$ for a polynomial poly. Our result in Shoup's model nearly matches this threshold by ruling out correctness probabilities as low as $\frac{3}{q} + \frac{1}{\mathsf{poly}}$. Also, note that when correctness is at least $\frac{1}{2} + \frac{1}{\mathsf{poly}}$, it can be amplified arbitrarily close to 1 via parallel repetition.

A conceptual framework for Shoup impossibilities. Proving impossibility results in Shoup's model is notoriously challenging, and as a result, nearly all prior results have been limited to Maurer's model (see [47]). Our proof introduces a novel strategy: We simulate any IPFE construction in

[1] Our impossibility results rule out both regimes of correctness. When q is very large (e.g., exponential), the bound $\frac{2}{q} + \frac{1}{q^{n-1}} + \frac{1}{\mathsf{poly}}$ yields a stronger impossibility. However, this bound becomes vacuous for $q = 2$, in which case the alternative bound $\frac{1}{2} + \frac{1}{\mathsf{poly}}$ provides a meaningful guarantee.

Shoup's model within an augmented version of Maurer's model, which enhances Maurer's with access to a random oracle (RO). This modular perspective enables us to decouple the two idealized models — Maurer's GGM and the RO — and apply our existing techniques for Maurer's model in conjunction with a recent combinatorial lemma used to prove the impossibility of IPFE in the RO model [27]. The main limitation of our approach — namely, the requirement that q^n be super-polynomial — arises directly from the bounds inherent to that combinatorial lemma.

Because no modular framework for Shoup-style lower bounds existed previously, most existing impossibility results were proved only in the more limited Maurer model, e.g., [18, 7, 22, 38, 36]. We believe the modular framework formalized in this work can facilitate lifting some of these results to Shoup's model.

Implications for multi-input quadratic FE. We observe that multi-input quadratic functional encryption (MI-QFE) [9] implies FH-IPFE. Given an MI-QFE scheme, one can embed the FH-IPFE's secret-key vectors f and the plaintext vector x as two different MI-QFE inputs and set the quadratic function F to compute their inner product (i.e., $F(f, x) = \langle f, x \rangle$). Concretely, $(\mathsf{sk}_F, \mathsf{ct}_f)$ becomes the functional secret key and ct_x the ciphertext in the FH-IPFE scheme. The semantic security of MI-QFE ensures that the resulting IPFE is function-hiding (Section G). Therefore, our impossibility result for FH-IPFE directly yields the following:

Informal Theorem 3. There does not exist a 2-input single-corruption non-compact quadratic functional encryption over $\mathbb{Z}_q^n$ with correctness probability ν in
(a) Maurer's GGM for the same restrictions on n, q and ν as in Informal Theorem 1, and
(b) Shoup's GGM for the same restrictions on n, q and ν as in Informal Theorem 2.

This result rules out the weakest possible form of MI-QFE: 2 inputs, a single corruption and no compactness requirements. Moreover, our result implies a separation between non-compact QFE and non-compact MI-QFE. The former (equivalent to non-function-hiding IPFE) can be constructed in the GGM, while we show the latter is GGM impossible. The above result shows that even the weakest variant of non-compact MI-QFE cannot be built from non-compact QFE in a black-box way. This is in contrast to the equivalence of IPFE and MI-IPFE [3]. Thus, our result also shows a separation between MI-IPFE and non-compact MI-QFE, which is in contrast to the equivalence of IPFE and non-compact QFE.

Implications for FE with polynomially many functions. When the set of supported functions $\mathcal{F}$ is of polynomial size, one can build 'vanilla' FE from minimal assumptions. We showed that this fails for function-hiding IPFE in Maurer's GGM. Moreover, this limitation is not confined to function hiding. By combining the above impossibility with Informal Theorem 3, we identify a second regime where the same obstruction arises, namely single-corruption 2-input QFE for arbitrary polynomial size functions is also impossible in Maurer's GGM.

Positive results for bounded collusion. Motivated by our negative results and by our demonstration of the non-triviality of *bounded collusions* in 2-input QFE, we investigate FH-IPFE under one-sided bounded collusions. Here, the adversary may obtain arbitrarily many ciphertexts, but the number of keys that can be corrupted is bounded by a fixed Q.[2] One can also define the dual

[2]Since FH-IPFE is defined in the private-key setting, the adversary obtains ciphertexts via an encryption oracle.

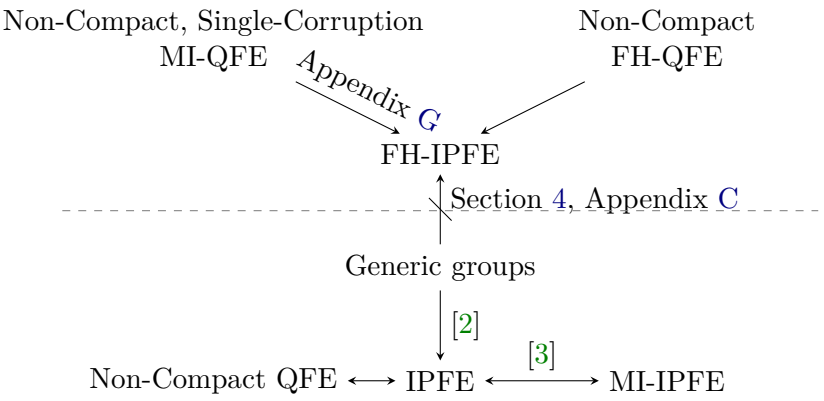

Figure 1: The relation between generic groups, IPFE, non-compact QFE, and its function-hiding and multi-input variants. Our main result is the separation of FH-IPFE from generic groups, which implies separations for all primitives above the dashed line from all primitives below the dashed line.

notion, where only the number of encryption queries is bounded. In this one-sided bounded model, we obtain the following result.

Informal Theorem 4. There exists a generic construction of one-sided bounded-collusion FH-IPFE from bounded-collusion (non-function-hiding) private-key IPFE, and hence from one-way functions.

Previously, it was only known how to boost an FE to its function-hiding variant if the original FE supports all functions computable by bounded-size circuits [20]. Notably, our reduction is both simple and concretely efficient. Combined with the generic construction of bounded-collusion private-key FE from OWFs [13], this yields a one-sided bounded-collusion FH-IPFE from OWFs. Alternatively, one can instantiate our reduction using DDH-based (unbounded collusion) IPFE, which may offer better concrete efficiency.

We also consider the two-sided bounded-collusion model, where both the number of functional secret keys and the number of ciphertexts are bounded. This weaker model suffices for several applications of FH-IPFE, such as bounded-collusion inner-product revealing encryption [6]. In this setting, we obtain:

Informal Theorem 5. There exists an information-theoretic FH-IPFE in the two-sided bounded-collusion model.

2 Technical Overview

An oracle-aided private-key IPFE for $\mathbb{Z}_q^n$ is described by $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$. The setup algorithm $\mathrm{Setup}^O(1^\kappa)$ outputs a master secret key msk. The algorithms KGen^O and Enc^O on input $(\mathsf{msk}, \mathbf{x} \in \mathbb{Z}_q^n)$ output, respectively, a functional key $\mathsf{sk}[\mathbf{x}]$ and a ciphertext $\mathsf{ct}[\mathbf{x}]$. The decryption algorithm Dec^O inputs pp, a functional key $\mathsf{sk}[\mathbf{y}]$, and a ciphertext $\mathsf{ct}[\mathbf{x}]$, and outputs the inner product $\langle \mathbf{y}, \mathbf{x} \rangle$.

The standard indistinguishability (FH-IND-CPA) notion says that a PPT adversary who makes LoR key-corruption queries $\{(\mathbf{y}_i, \mathbf{y}'_i)\}_i$ and LoR encryption queries $\{(\mathbf{x}_h, \mathbf{x}'_h)\}_h$, under the restriction

that $\langle \mathbf{y}_i, \mathbf{x}_h \rangle = \langle \mathbf{y}'_i, \mathbf{x}'_h \rangle$ for all values of i and h, should not be able to distinguish if the received responses correspond to the first component or the second one.

2.1 Impossibility in Maurer's GGM

We first present our impossibility result in Maurer's GGM.

Big picture: In a FH-IPFE, both ciphertexts $\mathsf{ct}[\mathbf{x}]$ and secret keys $\mathsf{sk}[\mathbf{y}]$ need to hide the vectors $\mathbf{x}$ and $\mathbf{y}$. In all existing pairing-based constructions, $\mathsf{ct}[\mathbf{x}]$ and $\mathsf{sk}[\mathbf{y}]$ contain group-based encryptions of $\mathbf{x}$ and $\mathbf{y}$, subsequently combined via pairings in decryption to recover $\langle \mathbf{x}, \mathbf{y} \rangle$.

We show that in the absence of pairings, not *both* $\mathrm{Enc}^{\mathbb{G}}$ and $\mathrm{KGen}^{\mathbb{G}}$ can use the group in a meaningful way to encrypt the vectors $\mathbf{x}$ and $\mathbf{y}$. More concretely,

1. We show how to compile a given GGM-based FH-IND-CPA IPFE $\mathcal{E}^{\mathbb{G}} = (\mathrm{Setup}^{\mathbb{G}}, \mathrm{KGen}^{\mathbb{G}}, \mathrm{Enc}^{\mathbb{G}}, \mathrm{Dec}^{\mathbb{G}})$ into $\overline{\mathcal{E}}^{\mathbb{G}} = (\overline{\mathrm{Setup}}^{\mathbb{G}}, \overline{\mathrm{KGen}}, \overline{\mathrm{Enc}}^{\mathbb{G}}, \overline{\mathrm{Dec}}^{\mathbb{G}})$ with 'comparable security and correctness', such that $\overline{\mathrm{KGen}}$ makes no $\mathbb{G}$ queries.

Intuitively, if $\overline{\mathrm{KGen}}$ does not use the generic group, it cannot encrypt the vector $\mathbf{y}$ in the GGM. We formalize this as follows.

2. We give a generic, information-theoretic attack against any FH-IPFE whose $\overline{\mathrm{KGen}}$ algorithm makes no oracle queries. Our attack works for every oracle, and not just a GGM oracle.

Challenges: Both Steps 1 and 2 above involve significant challenges. The main challenge in Step 1 is that we deal with a private-key IPFE scheme (as opposed to a public-key one), in which encryptions are made using msk, not known to the adversary. In the public-key setting (in which encryptions are made with a public mpk) there are compilation techniques [35, 38] that allow one to implement something similar to Step 1 for primitives such as Identity-Based Encryption (IBE). Those techniques rely crucially on an adversary seeing encryptions of many random plaintexts, a property inherent to public-key schemes but absent in the private-key setting. We introduce techniques for a compilation strategy adaptable to the private-key setting that preserves both correctness and a security property weaker than FH-IND-CPA, which nonetheless suffices for Step 2. Our novelty involves formalizing these new security definitions that will allow us to preserve both security and correctness. We elaborate more in Section 2.1.1.

The challenge behind Step 2 is that the FH-IPFE scheme may have a decryption error. Even starting from a perfectly correct scheme, the Step 1 compilation introduces a small error. It is relatively easy to do Step 2 for perfectly-correct schemes. But the strategy employed fails completely when we have decryption errors. We develop techniques to handle these errors (Section 2.1.2).

2.1.1 Step 1: The Compiler.

To describe the challenges and our techniques, let us review the compilation strategy of [35, 38], adapted to **public-key** IPFE and for non-FH IND-CPA. The latter is defined as in FH-IND-CPA (Section 2) except that for LoR key-corruption queries $(\mathbf{y}_i, \mathbf{y}'_i)$ we must always have $\mathbf{y}_i = \mathbf{y}'_i$.

Before describing this previous compilation, let us review the GGM model. In Maurer's GGM, the algorithms do not have explicit access to group elements. Instead, they get 'pointers' to group elements and can perform operations with the following two oracles: The Add oracle adds two group elements and gives a 'pointer' to the new group element. The Eql oracle compares two group

elements for equality. An algorithm that outputs group elements simply outputs pointers to group elements. If an algorithm outputs no such pointers, we say its output contains no group elements.

For algorithms $\mathsf{Alg}^{\mathsf{Add},\mathsf{Eql}}(g_1,\ldots,g_m)$ that do not output any group elements, we can remove access to both Add and Eql oracles if we provide a stronger Eql oracle, called Eql^*. An Eql^* query takes $(a_1,\ldots,a_m,g_1,\ldots,g_m)$ and outputs 1 iff $a_1g_1+\cdots+a_mg_m=0$. Now we can simulate Add queries of Alg as follows. We return to A a new 'virtual' group element, which is a pointer to a 'dummy' group element. For each virtual group element we store the inputs of the Add query. This way we know for every virtual group element g coefficients $a_1,\ldots,a_m$ such that $g=a_1\cdot g+\cdots+a_mg_m$, where $g_1,\ldots,g_m$ are the group elements given to Alg as input. When the virtual group element is later used in an Eql query, we can replace it with its linear combination of input group elements and make an equivalent Eql^* query. Note that introducing virtual group elements works precisely because Alg does not output any group elements itself.

Compilation of [35, 38]: These compilers, discussed here for non-function-hiding **public-key** IPFE, start by removing all group elements from functional keys, while preserving (a form of) correctness and security. Once this is done, it is easy to eliminate any $\mathbb{G}$ queries made during KGen.

We describe this latter step first. Since KGen does not output any group elements (as functional keys contain no group elements), by the discussion above, we may assume KGen makes only Eql^* queries. We can let msk store the discrete logarithms of all group elements generated during Setup. Since KGen takes msk as input, knowledge of these stored discrete logarithms allows KGen to answer all issued Eql^* queries without calling the Eql^* oracle. Because msk is never given to the adversary, this does not compromise security.

We now now discuss removing group elements from functional keys by recalling the compilations of [35, 38] adapted to *public key, non-function hiding* IPFE. Let $\mathsf{sk}[\mathbf{y}]:=(s\in\{0,1\}^*,g_1,\ldots,g_t)$ be a functional key with group elements $g_1,\ldots,g_t$. We describe how to obtain a key $\overline{\mathsf{sk}[\mathbf{y}]}$ from $\mathsf{sk}[\mathbf{y}]$ that does not contain any group elements.

To have fewer variables, suppose each mpk and sk have t group elements. Also, suppose $\mathrm{Dec}(\mathsf{sk}[\mathbf{y}],\mathsf{ct}[\mathbf{x}])$ outputs the inner product $\langle\mathbf{x},\mathbf{y}\rangle$ directly. We sketch at the end of this section how to handle IPFE schemes where Dec just outputs $(\langle\mathbf{x},\mathbf{y}\rangle)\cdot g$, which is typically the case for group-based (FH-)IPFE schemes.

1. For $i\in[\ell]$, sample $\mathbf{x}_i\leftarrow\mathbb{Z}_q^n$, and run $\mathrm{Dec}(\mathsf{sk}[\mathbf{y}],\mathsf{ct}_i:=\mathrm{Enc}(\mathsf{mpk},\mathbf{x}_i))$. Let $u_1,\ldots,u_t$ be the group elements of mpk. Since running Enc and then Dec together does not output any group elements, we can remove all Add queries and accept instead Eql^* queries, which are always over $(g_1,\ldots,g_t,u_1,\ldots,u_t)$, as described above. For each positively answered Eql^* query, i.e., $\sum_{i=1}^t a_i\cdot g_i+\sum_{i=1}^t a'_iu_i=0$, we add the coefficients $(a_1,\ldots,a_t,a'_1,\ldots,a'_t)$ as a new row to a matrix R. The compiled secret key $\overline{\mathsf{sk}[\mathbf{y}]}=(s,R)$ is then obtained by removing all the group elements from $\mathsf{sk}[\mathbf{y}]$, and augmenting it with R.
2. For a given ciphertext $\mathsf{ct}=(w\in\{0,1\}^*,z_1,\ldots,z_t)$, we simulate the execution of $\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}],\mathsf{ct})$ using $\overline{\mathsf{sk}[\mathbf{y}]}$ as follows. Since Dec outputs no group element (but only an element in $\mathbb{Z}_q$) we may assume it only makes Eql^* queries. For a given such query

$$\sum_{i=1}^t a_ig_i+\sum_{i=1}^t a'_iu_i+\sum_{i=1}^t a''_iz_i\stackrel{?}{=}0, \tag{1}$$

the compiled $\overline{\mathsf{sk}[\mathbf{y}]}$ does not have explicit access to $(g_1,\ldots,g_t)$, since those group elements where remove from $\overline{\mathsf{sk}[\mathbf{y}]}$. Instead, $\overline{\mathrm{Dec}}$ attempts reducing the relation check of Equation 1 into

an equivalent one involving only $\{u_i\}$ and $\{z_i\}$, both of which it has explicit access to. To this end, $\overline{\mathsf{Dec}}$ checks if $(a_1, \ldots, a_t) \in \mathsf{Span}(R')$, where R' is the submatrix of R that contains only those columns corresponding to $(g_1, \ldots, g_t)$. If $(a_1, \ldots, a_t) \notin \mathsf{Span}(R')$ (call this Condition (*)), then $\overline{\mathsf{Dec}}$ responds to Equation 1 with 0. Else, $\overline{\mathsf{Dec}}$ obtains an equivalent relation that involves only $\{u_i\}$ and $\{z_i\}$, which it can query Eql^* on.

Correctness follows by setting ℓ large enough. Specifically, $\overline{\mathsf{Dec}}$ makes a mistake only when Condition (*) occurs but the answer to Equation 1 is 1. Call this the bad case. By using the Chernoff bound and the fact that each $\mathbf{x}_i$ is chosen uniformly at random, one may argue that when decrypting the encryption of a random $\mathbf{x}$ with $\overline{\mathsf{sk}[\mathbf{y}]}$, the probability that the bad case happens is small.

For non-FH IND-CPA, one should argue $\overline{\mathsf{sk}[\mathbf{y}]}$ can be simulated from $\mathsf{sk}[\mathbf{y}]$ by an adversary with full access to $\mathbb{G}$. This follows immediately, since the augmentation involves sampling many ciphertexts of random vectors, which can be simulated knowing only mpk. This is where things break down in the private-key setting.

Take-home summary: What makes the above compilation [35, 38] possible is that each secret key $\mathsf{sk}[\mathbf{y}]$ is compiled against a list $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ of ciphertexts each encrypting a **uniformly random vector**. This is needed for correctness. And the resulting augmentation does not harm security because all those encryptions can be obtained for free (i.e., from mpk).

The Compiler [35, 38] for FH-IPFE? The main issue in applying the above compiler to FH-IPFE is that it destroys FH-IND-CPA security. To see this, note that a given $\mathsf{sk}[\mathbf{y}]$ is compiled by performing many encryptions of random vectors $\mathbf{x} \leftarrow \mathbb{Z}_q^n$. To ensure security is preserved, we have to show that an adversary can simulate $\overline{\mathsf{sk}[\mathbf{y}]}$ from $\mathsf{sk}[\mathbf{y}]$. To do this, the adversary should be given many encryptions of random plaintexts $\mathbf{x}$ from $\mathbb{Z}_q^n$. But this cannot be provided in general under the FH-IND-CPA game for private-key IPFE. In particular, imagine an adversary $\overline{\mathcal{A}}$ against the compiled scheme that makes a single LoR key corruption query for $(0^n, 1^n)$, and no other queries. To simulate the view of $\overline{\mathcal{A}}$ by an adversary $\mathcal{A}$ against the base scheme, $\mathcal{A}$ must also make the same query to get a secret key sk encoding either 0^n and 1^n. To compile this key sk, $\mathcal{A}$ must be given many encryptions of random vectors $\mathbf{x}$ from $\mathbb{Z}_q^n$. However, $\mathcal{A}$ cannot obtain these ciphertexts via encryption queries, because $\mathcal{A}$'s queries will become inadmissible (i.e., w.h.p. $\langle 0^n, \mathbf{x}\rangle \neq \langle 1^n, \mathbf{x}\rangle$ for a random $\mathbf{x}$). This was not an issue for public-key IPFE with non-FH IND-CPA because encryptions there were free.

The above compiler, which preserves correctness but not security, illustrates the challenges in getting both correctness and security. Looking ahead, our solution is a careful weakening of FH IND-CPA and a modified compilation such that (1) both security and correctness are preserved and (b) the new security notion is still strong enough to be broken information-theoretically for schemes where KGen makes no queries.

First attempt. To give intuition about our final compilation, let us give a procedure which preserves a weaker security notion, but not correctness — the opposite of the above situation. Our weakened security notion (call it key hiding, formalized in Definition 3.3) is defined with respect to an $(n-1)$-dimensional subspace $\mathbf{V}$. The adversary may make key-corruption and encryption queries, but the adversary does not provide an inputs for any query. In response to a key-corruption query, the challenger samples $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{w} \leftarrow \mathsf{V}^\perp$, and returns $(\mathbf{y}, \mathbf{y}+\mathbf{w}, \mathsf{sk}[\mathbf{y}+b\mathbf{w}])$ to the adversary,

where b is the challenge bit and $\mathsf{V}^\perp$ is the orthogonal complement of V. In response to an encryption query, the challenger returns $(\mathbf{x}, \mathsf{ct}[\mathbf{x}])$, where $\mathbf{x} \leftarrow \mathsf{V}$ and $\mathsf{ct}[\mathbf{x}]$ is an encryption of $\mathbf{x}$.

We slightly modify the previous compilation as follows. To compile a secret key $\mathsf{sk} := (s, g_1, \ldots, g_m)$ into $\overline{\mathsf{sk}}$, and specifically to learn an underlying relation R among $\{g_i\}$ to be included in $\overline{\mathsf{sk}}$, we proceed as before by performing decryptions of many ciphertexts but now each ciphertext ct_i is encrypting a random vector from V, as opposed to from $\mathbb{Z}_q^n$. Namely, during decryption of each ct_i, we add to R all positively-answered equality relations. It is easy to argue that this transformation preserves key-hiding security because any adversary $\overline{\mathcal{A}}$ against the key-hiding of the compiled scheme can be simulated by an $\mathcal{A}$ against the base scheme. This is because encryption queries can be relayed from $\overline{\mathcal{A}}$ to $\mathcal{A}$. Moreover, for any key-corruption query of $\overline{\mathcal{A}}$, $\mathcal{A}$ may make a key-corruption query to get $(\mathbf{y}, \mathbf{y} + b\mathbf{w}, \mathsf{sk}[\mathbf{y} + b\mathbf{w}])$ and augment $\mathsf{sk}[\mathbf{y} + b\mathbf{w}]$ by requesting many encryption queries (resulting in ciphertexts encrypting random vectors in V) and decrypting them with $\mathsf{sk}[\mathbf{y} + b\mathbf{w}]$. Moreover, it is easy to argue correctness as long as plaintext vectors are chosen uniformly at random from V, as opposed to the entire space $\mathbb{Z}_q^n$. Call this weak correctness.

We may hope that the key-hiding property of any SK-IPFE whose KGen algorithm makes no $\mathbb{G}$ queries (as the compiled scheme above) and which provides weak correctness can be broken information-theoretically, so to reach a contradiction. However, this intuition is completely false. In fact, it is possible to build SK-IPFE which provides weak correctness as above and key hiding, information theoretically! Let $\mathrm{KGen}(\mathsf{msk}, \mathbf{y})$ output $\mathbf{w} + \mathbf{y}$ for a random $\mathbf{w} \leftarrow \mathsf{V}^\perp$. Let $\mathrm{Enc}(\mathsf{msk}, \mathbf{x})$ return $\mathbf{x}$. The decryption algorithm simply outputs the inner product. It is easy to argue that this trivial scheme satisfies both weak correctness and key hiding.

Fixing the above issue. We point out another issue with the above approach. Note that our key-hiding definition says nothing about plaintext hiding, and so it can be realized from the DDH assumption (even) with perfect correctness. (If we require weak correctness in the sense of above as opposed to perfect correctness, then as argued above, the notion can be realized information theoretically.) Specifically, the dual of any non-FH IND-CPA IPFE (e.g., [2]) satisfies key hiding, where by dual we mean swapping the roles of KGen and Enc. Thus, we cannot rule out perfectly-correct key-hiding SK-IPFE schemes in the GGM model. Note, however, that we have not yet made use of any plaintext hiding properties, which must also be satisfied by FH-IPFE schemes. We formalize a plaintext-hiding property and show that any perfectly-correct (or even $(1 - \mathrm{negl})$ correct) SK-IPFE with both plaintext hiding and key hiding properties can be transformed into a SK-IPFE $\overline{\mathcal{E}}$ which is almost as correct, and satisfies the key-hiding property and whose $\overline{\mathrm{KGen}}$ makes no $\mathbb{G}$ queries. We then break such schemes information theoretically. Note that breaking such SK-IPFE schemes with key hiding does not contradict the above positive result, because the above positive result (the dual of [2]) results in KGen making $\mathbb{G}$ calls.

The plaintext hiding property (formalized in Definition 3.4) is defined as follows. Imagine an adversary $\mathcal{A}$ that is given $\mathsf{sk}[\mathbf{y}]$ for an arbitrary $\mathbf{y} \notin \mathsf{V}^\perp$, and is given ℓ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, where each ct_i is either encrypting a random vector from $\mathbf{V}$ (if $b = 0$) or from $\mathbb{Z}_q^n$ (if $b = 1$). Suppose $\mathcal{A}$ is also given an encryption of a random vector $\mathbf{x}$ from $\mathbb{Z}_q^n$, together with $\langle \mathbf{y}, \mathbf{x} \rangle$. We show that FH-IND-CPA implies the challenge bit remains hidden. Suppose the original scheme $\mathcal{E}$ is perfectly correct, and provides plaintext hiding. (Again we can relax this to $(1 - \mathrm{negl})$ correctness.) We will show that the compiled scheme has $(1 - \frac{1}{\mathsf{poly}})$ correctness for every choice of a secret key vector $\mathbf{y} \notin \mathsf{V}^\perp$ (i.e., for every such $\mathbf{y} \notin \mathsf{V}^\perp$, if we sample $\overline{\mathsf{sk}[\mathbf{y}]}$ and $\mathsf{ct}[\mathbf{x}]$ for a random $\mathbf{x}$, $\overline{\mathrm{Dec}}(\overline{\mathsf{sk}[\mathbf{y}]}, \mathsf{ct}[\mathbf{x}])$ outputs $\langle \mathbf{y}, \mathbf{x} \rangle$ with probability close to one). This implies almost-always correctness because $\mathsf{V}^\perp$

has a small weight over $\mathbb{Z}_q^n$. Fix $\mathbf{y} \notin \mathsf{V}^\perp$ and let α be the probability of correctness under $\mathbf{y}$ for the compiled scheme, as defined above. Suppose $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ given above are used for the compilation procedure.

1. If $b = 0$, namely all ct_i are encrypting V-vectors, then the probability that $\overline{\mathrm{Dec}}$ correctly decrypts ct to $\langle \mathbf{y}, \mathbf{x} \rangle$ is α. This follows because in our compilation above, we compile secret keys against V ciphertexts.
2. If $b = 1$, namely all ct_i are encrypting fully-random vectors, then the probability that $\overline{\mathrm{Dec}}$ correctly decrypts ct to $\langle \mathbf{y}, \mathbf{x} \rangle$ is $\beta \approx 1$. This follows from the techniques of [35, 38], explained before.
3. By plaintext hiding $\alpha \approx \beta$, because otherwise an adversary can distinguish $b = 0$ from $b = 1$; i.e., compile $\mathsf{sk}[\mathbf{y}]$ against $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, decrypt ct using the compiled secret key and output 1 iff the result equals $\langle \mathbf{y}, \mathbf{x} \rangle$.

We have now argued the compiled scheme has key hiding, has correctness ≈ 1, and has a $\overline{\mathrm{KGen}}$ algorithm that makes no oracle queries. We break such schemes in Section 2.1.2.

Learning the inner product in the exponent. So far, we have assumed that the decryption algorithm $\mathrm{Dec}^{\mathbb{G}}$ outputs the inner product in the clear, i.e., as a $\mathbb{Z}_p$ element. Accordingly, we assumed that $\mathrm{Dec}^{\mathbb{G}}$ makes no Add queries and only issues Eql queries. However, decryption in existing GGM-based IPFE schemes (e.g., [2]) does not output the inner product itself, but rather an encoding of it, such as $(\langle \mathbf{y}, \mathbf{x} \rangle) \cdot g$, where g is a generator of the group. To capture this setting, we modify our model by allowing $\mathrm{Dec}^{\mathbb{G}}$ to take as input a candidate inner product value z and output 0/1. Concretely, we view Dec as operating in two phases. In the first phase, $\mathrm{Dec}_0^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}])$ outputs a single group element $\widetilde{g}$. In the second phase, $\mathrm{Dec}_1(g, \widetilde{g}, z)$ outputs 0/1, corresponding to the equality test $z \cdot g =^? \widetilde{g}$. We show how to extend our compilation to handle this more general setting. More generally, one could define richer variants of $(\mathrm{Dec}_0, \mathrm{Dec}_1)$ in which, for example, Dec_0 outputs multiple group elements and Dec_1 performs equality checks on linear combinations of these elements together with $z \cdot g$. For clarity of exposition, however, we restrict attention to the above $(\mathrm{Dec}_0, \mathrm{Dec}_1)$ formulation.

2.1.2 Step 2: The Information-Theoretic Attack.

We now prove Step 2. Specifically, we show how to break the key-hiding property of any $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}, \mathrm{Enc}^O, \mathrm{Dec}^O)$, with an O-free algorithm KGen, which has correctness ≈ 1. Interestingly, our attacker $\mathcal{A}$ does not need to make any O queries at all. Recall that under the key-hiding game, $\mathcal{A}$ has to guess the challenge bit b, where $\mathcal{A}$ is given $(\mathbf{y}_i, \mathbf{z}_i = \mathbf{y}_i + \mathbf{w}_i, \mathsf{sk}[\mathbf{y}_i + b\mathbf{w}_i])$ for $i \in [\ell]$, where $\mathbf{y}_i \leftarrow \mathbb{Z}_q^n$, $\mathbf{w}_i \leftarrow \mathsf{V}^\perp$ and V is an $(n-1)$ dimensional subspace. Also, ℓ is an arbitrary polynomial. The key-hiding game also allows for encryption queries, but $\mathcal{A}$ does not need to make such queries.

As a warm-up assume $\mathcal{E}^O$ is perfectly correct. The attack in this case is simple. But it is not sufficient for our general purpose because our compilation induces a small decryption error.

In a perfectly correct IPFE $\mathcal{E}^O$, given a fixed master secret key msk generated by Setup^O, each $\mathsf{sk}[\mathbf{y}]$ is a perfect commitment to $\mathbf{y}$; i.e, there does not exist $\mathbf{y}$ and $\mathbf{z}$ such that $\mathbf{y} \neq \mathbf{z}$ but a secret key sk is in the image of both $\mathrm{KGen}(\mathsf{msk}, \mathbf{y})$ and $\mathrm{KGen}(\mathsf{msk}, \mathbf{z})$. With this intuition in mind, $\mathcal{A}$ determines the value of b from $\{(\mathbf{y}_i, \mathbf{z}_i = \mathbf{y}_i + \mathbf{w}_i, \mathsf{sk}_i)\}$ as follows. We will write $x \in \mathcal{D}$ for a distribution $\mathcal{D}$ to mean x is in the support of $\mathcal{D}$.

- If there exists $\widetilde{\mathsf{msk}}$ such that $\mathrm{Recover}(\widetilde{\mathsf{msk}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) = (\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$, return 0. (We define the procedure Recover momentarily.) Else, return 1.

The procedure $\mathrm{Recover}(\widetilde{\mathsf{msk}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$ works as follows: if there exists $(\mathbf{f}_1, \ldots, \mathbf{f}_\ell)$ such that for all $i \in [\ell]$, $\mathsf{sk}_i \in \mathrm{KGen}(\widetilde{\mathsf{msk}}, \mathbf{f}_i)$ and $\mathsf{sk}_i \notin \mathrm{KGen}(\widetilde{\mathsf{msk}}, \mathbf{f}'_i)$ for all $\mathbf{f}'_i \neq \mathbf{f}_i$, return $(\mathbf{f}_1, \ldots, \mathbf{f}_\ell)$. Else, return $\perp$.

Note that $\mathcal{A}$ makes no O queries; in particular, the procedure Recover involves making no O queries.

It is easy to see that whenever $b = 0$, $\mathcal{A}$ outputs 0 with probability one. We now argue that if $b = 1$, then $\mathcal{A}$ outputs 0 with negligible probability, assuming ℓ is large enough. This will complete the proof.

Suppose $b = 1$, meaning that each sk_i is encoding $\mathbf{z}_i = \mathbf{y}_i + \mathbf{w}_i$. Given $\mathbf{z}_i$ the probability one can guess the value of $\mathbf{y}_i$ is at most $\frac{1}{|\mathsf{V}^\perp|} = \frac{1}{q}$. Thus, for any fixed $\widetilde{\mathsf{msk}}$, the probability that $\mathrm{Recover}(\widetilde{\mathsf{msk}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$ outputs $(\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$ is at most $\frac{1}{q^\ell}$. Assuming $m = |\mathsf{msk}|$, by choosing ℓ such that $\frac{2^m}{q^\ell} = \mathrm{negl}(\kappa)$, we get the desired result.

Going beyond perfect correctness. If $\mathcal{E}^O$ has even a negligible correctness error, the above attack completely fails. In fact, not a single functional key might be perfectly binding in the above sense. To see why, change a perfectly-correct IPFE as follows: when generating a key for $\mathbf{y}$ under msk, with probability $1 - \frac{1}{q^n}$, generate a key for $\mathbf{y}$ itself, and with probability $\frac{1}{q^n(q^n-1)}$ generate a key for $\mathbf{y} + \mathbf{s}$ for any non-zero shift $\mathbf{s}$. Here for every functional secret key we will have exponentially many pre-images. As such, designing an attack in this case is delicate, requiring one to come up with the right notion of 'binding'.

We define the new binding probability as follows. We say $(\mathsf{msk}, \mathsf{sk})$ is binding to $\mathbf{y}$ if by reverse sampling a secret-key vector based on pre-images of sk we get $\mathbf{y}$ with a probability strictly greater than $1/2$. That is,

$$\Pr_{\mathbf{x} \leftarrow \mathbb{Z}_q^n}[\mathbf{x} = \mathbf{y} \mid \mathrm{KGen}(\mathsf{msk}, \mathbf{x}) = \mathsf{sk}] > \frac{1}{2}. \tag{2}$$

Note that given msk, each sk binds itself to at most one secret-key vector. And note that, since KGen is oracle free, one can find the vector $\mathbf{y}$ (if any) to which sk is binding, without making any O queries.

Suppose for simplicity that every msk output by Setup induces a decryption error of at most 0.001. (Our actual proof will relax this assumption.) We will show that with high probability (greater than 33/60), $(\mathsf{msk}, \mathsf{sk}[\mathbf{y}])$ will be a binding to $\mathbf{y}$ (Equation 2), where $\mathbf{y}$ is a random vector and $\mathsf{sk}[\mathbf{y}]$ is a random secret key for $\mathbf{y}$. We will prove this by showing that if this does not hold, the probability of correctness will be strictly less than 0.999. With this in mind, we may then change the procedure Recover above so that for a given $\widetilde{\mathsf{msk}}$ and for each component i, it recovers the unique vector $\mathbf{h}_i$ to which sk_i is binding (if any), and outputs 0 if $(\mathbf{h}_1, \ldots, \mathbf{h}_\ell)$ agrees with $(\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$ on a large fraction of coordinates (close to 33/60 above).

Our final proof is delicate, formalizing all of the above probability conditions.

2.2 Impossibility in Shoup's GGM

Maurer's GGM is restrictive, and several cryptographic primitives, such as delay functions [36], signatures [24], compact encryption, and several other primitives [47] are impossible in Maurer's

GGM, even though they can be realized from concrete groups. Informally, the lack of a bit representation for group elements is what makes Maurer's GGM restrictive.

In contrast, Shoup's GGM [40], where each group element is represented by a unique random bitstring (the "label"), does not suffer from this lack of expressiveness. In fact, all the primitives mentioned above can be realized in Shoup's model. Thus, an impossibility result in Shoup's model is a much stronger indication that a primitive cannot be built from concrete groups, such as elliptic curves. Unfortunately, proving impossibility results in Shoup's model is also much more challenging, and as such almost all existing impossibility results were proved only in Maurer's model (e.g. [18, 7, 47, 22, 24, 38, 36]). For example, the impossibility of IBE in Maurer's GGM [35, 38] was only generalized recently to Shoup's model [47], using very different techniques.[3]

For our impossibility result for FH-IPFE in Maurer's GGM, outlined earlier, it may seem the techniques cannot be applied in Shoup's model. Recall that our techniques relied on removing group elements from the secret key while keeping the bitstring part. In Shoup's model, however, each group element has a bit representation, so it is not even well-defined which part of a secret key is a group element and which part is a bitstring that does not encode a group element.

Nevertheless, we show that our impossibility result in Maurer's GGM can in fact be extended to Shoup's GGM *in a modular way*. We believe that our techniques can be adapted to lift other impossibility results from Maurer's GGM to Shoup's GGM in a similarly modular fashion.

2.2.1 Maurer's GGM with Random Representations.

For our modular approach we introduce a new model that we call *Maurer's GGM with random representations*. This model is defined like Maurer's GGM model with the oracles $\mathbb{G} = (\mathsf{Add}, \mathsf{Eql})$, but additionally each group element is assigned a random bitstring label, from a set $\mathsf{LB} \subseteq \{0,1\}^*$. Concretely, we add an oracle $\mathsf{PointerToLabel}$ that inputs a group element, which is represented via a pointer, as in Maurer's model, and outputs its assigned label as an ordinary bitstring. We also add an oracle $\mathsf{LabelToPointer}$ that is the inverse of $\mathsf{PointerToLabel}$: It inputs a bitstring label and returns the group element that was assigned this label, represented as pointer, or a failure symbol $\perp$ if the bitstring is not a valid label of any group element. We use $\mathbb{L} := (\mathsf{PointerToLabel}, \mathsf{LabelToPointer})$ to denote both of these oracles. We call this model the random label Maurer GGM (RL-MGGM) model and an algorithm in this model has oracle access to both $\mathbb{G}$ and $\mathbb{L}$ (Definition B.2).

In this model it is easy to simulate Shoup's generic group model, because we can use the $\mathbb{L}$ oracle to obtain a random bit representation for each group element and also convert these bit representations back to 'pointer-type' group elements. More specifically, Shoup's model can be described by a single oracle SAdd that inputs two labels of group elements and outputs the label of the sum of these group elements. We can simulate this by converting these labels to pointers to group elements via the $\mathsf{LabelToPointer}$ oracle, compute their sum with the Add oracle of Maurer's model, and convert the resulting group element pointer back to a label with the $\mathsf{PointerToLabel}$ oracle.

This model might seem like an overly complicated description of Shoup's GGM. However, it has the advantage that it separates the algebraic structure of groups from the random labels. On a high level, we will use the technique for the separation in Maurer's model to handle the $\mathbb{G}$ queries of the RL-MGGM model. The $\mathbb{L}$ oracle is essentially a random permutation, that can be handled using techniques to separate SK-IPFE from random permutations from prior work [27, 28].

[3]The work [35] claims that their impossibility is in Shoup's model. However, as noted in [47], they impose GGM restrictions that are only allowed in Maurer's model.

However, even with this model and these techniques the problem is still non-trivial. Remember that the first step of our impossibility result is to remove all oracle queries from KGen. To achieve this, we remove group elements from functional keys and ensure that KGen knows the discrete log of every group element in its input. Then, KGen can be simulated without ever obtaining a pointer to a group element and therefore without any oracle queries. This argument completely fails in this new model, because KGen might use pointers to group elements as input to the $\mathsf{PointerToLabel}$ oracle. Concretely, imagine that KGen samples $z \leftarrow \mathbb{Z}_q$ and includes $(z, \mathsf{Label} := \mathsf{PointerToLabel}(z \cdot g))$ in the functional key. Dec checks if $\mathsf{Label} := \mathsf{PointerToLabel}(z \cdot g)$ holds and outputs a failure symbol if this check does not pass. If KGen has no oracle access, it is impossible to produce a pair (z, Label) that passes this check.

Challenge 1: KGen needs pointers to group elements (and therefore access to the oracle $\mathbb{G}$) to compute bit representations via the $\mathsf{PointerToLabel}$ oracle.

However, even if we would successfully remove all $\mathbb{G}$ queries from KGen (and, for example, would allow KGen retrieving the random representations providing just the exponent of the group element), we cannot use the information-theoretic attack outlined above, because this attack relies on KGen making no queries at all, and KGen still has oracle access to $\mathbb{L}$. A natural idea would be to remove the oracle $\mathbb{L}$ from KGen, too. However, this seems to be impossible. For example, consider an IPFE that stores a random exponent z in the master secret key. The $\mathsf{Enc}(\mathsf{msk}, \mathbf{x})$ algorithm first creates an IPFE ciphertext $\mathsf{ct}[\mathbf{x}]$ for $\mathbf{x}$ and then additionally secret-key encrypts $\mathsf{ct}[\mathbf{x}]$ using the bit representation of $z \cdot g$ as secret key. Moreover, it adds a secret key encryption of the entire plaintext $\mathbf{x}$ under the secret key z to the ciphertext. All functional secret keys of this IPFE contain the bit representation of $z \cdot g$, so that decryption can recover the actual IPFE ciphertext $\mathsf{ct}[\mathbf{x}]$ from the SKE ciphertext. In this example, it is unclear how we can remove the $\mathbb{L}$ oracle from KGen. Adding the exponent z to the secret keys would violate security, because they can be used to recover the entire plaintext. But without adding any information about z, Dec will not be able to recover the IPFE ciphertext $\mathsf{ct}[\mathbf{x}]$, and thus we would violate correctness.

Challenge 2: It seems impossible to remove $\mathbb{L}$ queries from KGen.

Our proof strategy. To overcome Challenge 2, we introduce an alternative information-theoretic attack to the one outlined above. This alternative information-theoretic attack does not require us to remove all queries from KGen. Instead, it only requires that KGen and Dec each get access to different, independent oracles. This provides an alternative path to solve the second challenge.

Alternative challenge 2: Remove $\mathbb{L}$ queries from Dec.

The first step of our proof solves Alternative Challenge 2 by introducing a generic compiler that removes $\mathbb{L}$ queries from Dec. The second step of our proof solves Challenge 1 by replacing the $\mathbb{L}$ oracle with an ordinary random oracle. After these two compilation steps, we can use the compiler we have introduced for the Maurer impossibility result to remove $\mathbb{G}$ queries from KGen. Finally, we apply our new information-theoretic attack.

2.2.2 Step 1: Removing $\mathbb{L}$ Queries from Dec.

The idea to remove $\mathbb{L}$ queries from Dec is to simulate the random labels of group elements. Concretely, Dec keeps a table of group element handles and their corresponding label. Whenever a random

representation for a group element is queried, we look for this group element in the table and return its label, if we found it. If the table has no matching entry, we sample a new label (that is also stored in the table) and return it. Queries to convert random representations back to handles are handled similarly.

The obvious difficulty with that approach is that we need to ensure that we answer these queries consistently with those queries already made during generation of the ciphertext and the functional key. We cannot simply include the handle/label pairs that were used during encryption or key generation in the ciphertext or key, because that might violate security (for example, Enc and KGen could query a group element whose random representation matches the master secret key). Instead, we generate in KGen queries many ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_\mathsf{e}}$ that we decrypt and append the handle/label pairs that we collect during these decryptions to the secret keys. For Enc, we proceed analogously with many random functional keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{\ell_\mathsf{k}}$.

Now we face a similar challenge to the compiler for Maurer's GGM. To preserve correctness (for uniformly random vectors), we should sample the plaintexts for the ciphertexts uniformly at random from the entire space $\mathbb{Z}_q^n$. However, to argue security, we can only afford to sample them from a subspace $\mathsf{V} \subsetneq \mathbb{Z}_q^n$. In the compiler for Maurer's GGM we solved this by leveraging the plaintext hiding property of the scheme to argue that correctness is still preserved, when the ciphertexts come only from a subspace. While this argument worked for learning the relations among group elements, it does not work for learning group element labels.

To overcome this problem, we use a combinatorial trick that was introduced in [27] that essentially shows that we can learn the same query-answer pairs (for any oracle) if the vectors are sampled from a subspace (instead of the entire space). However, using this lemma comes at the price that we have to assume that q^n is superpolynomial (and $n \geqslant 4$). We leave it as an open problem to provide a positive or negative result for FH-IPFE in Shoup's GGM when q^n is only of polynomial size.

2.2.3 Step 2: Replacing $\mathbb{L}$ by a Random Oracle.

The above compilation is not only a necessity for the final information-theoretic attack on the scheme, it also makes the second step, replacing $\mathbb{L}$ by a random oracle, easy. We can assume that during Enc and KGen we know the discrete log of every group element. Thus, we can use the random oracle to build a pseudorandom permutation (PRP) [32]. We can then apply the PRP to the exponent of a group element to get the label and the inverse to get from a label the exponent.[4] This way, we can ensure that KGen and Enc generate all labels consistently. Since Dec makes no queries, correctness is preserved, even when the labels no longer match the labels of the oracle $\mathbb{L}$.

To argue that this compiler preserves security, we just have to reduce an adversary for the FH-IPFE where labels in Enc and KGen are inconsistent with the oracle $\mathbb{L}$ to one where they are consistent with the oracle $\mathbb{L}$. This can be achieved by a reduction that simulates the oracle $\mathbb{L}$ for the adversary independently of the real $\mathbb{L}$ and forwards all other queries.

2.2.4 Step 3: Remove $\mathbb{G}$ queries from KGen.

The next step is to run the compiler we already used for the impossibility result in Maurer's GGM to remove $\mathbb{G}$ queries from KGen.

[4] If we want the labels to be sparse, we interpret all labels that are mapped to an exponent larger than the group order as invalid labels.

Recall that this compiler relies on the plaintext hiding property of the FH-IPFE to argue that it preserves correctness. One difficulty is that the plaintext hiding notion we can preserve in the previous step is weaker than the one we used for the impossibility result in Maurer's model, because we have to make sure that our notion allows us to generate keys from a subspace $\mathsf{U} \subseteq \mathbb{Z}_q^n$ to be preserved in Step 1. This makes arguing correctness more challenging, and we provide an alternative proof that this notion still sufficient to guarantee that correctness is preserved in the compilation.

An advantage of this, more involved, argument, compared to the previous one, is that we can tune the decryption failures to an arbitrary small polynomial, which even improves our result for Maurer's model.

2.2.5 Step 4: Alternative Information-theoretic Attack.

After executing the compiler from step 1, 2, and 3 we end up with an FH-IPFE scheme where KGen makes no $\mathbb{G}$ queries, but it can still make random oracle queries. As mentioned above, this prevents us to use the previous, information-theoretic attack.

Instead, we leverage that KGen makes random oracle queries, but no $\mathbb{G}$ queries and Dec makes $\mathbb{G}$ queries, but not random oracle queries.

Our information-theoretic attack makes many secret key queries that return $(\mathbf{y}_i, \mathbf{y}'_i = \mathbf{y}_i + \mathbf{v}_i, \mathsf{sk}_i)$, where $\mathbf{y}_i \leftarrow \mathbb{Z}_q^n$, $\mathbf{v}_i \in \mathsf{V}^\perp$ and sk_i is a secret key for $\mathbf{y}_i + b\mathbf{v}_i$, where $b \leftarrow \{0,1\}$ is a challenge bit sampled once for all queries. The attack will make a brute force search for a master secret key msk, a vector $\mathbf{x} \notin \mathsf{V}$, a ciphertext $\mathsf{ct}[\mathbf{x}]$ encrypting $\mathbf{x}$ and a set of query-answer pairs of all queries made during the generation of msk and $\mathsf{ct}[\mathbf{x}]$. If decrypting $\mathsf{ct}[\mathbf{x}]$ with sk_i results in $\langle \mathbf{x}, \mathbf{y}_i \rangle$ for a large enough fraction of indices i, the adversary guesses $b = 0$ and otherwise $b = 1$. All queries made during decryption are simulated, which is possible for the adversary, since it knows the discrete log of every group element. Note that this simulation is faithful, because no $\mathbb{G}$ queries where made during the generation of the secret keys sk_i.

By carefully tuning the threshold of the number of indices for which decryption resulted in $\langle \mathbf{x}, \mathbf{y}_i \rangle$ and making the number of secret keys large enough, we can guarantee that this adversary wins with high probability.

For this information-theoretic attack, we need at least $q \geqslant 3$, but for $q \geqslant 5$ it even rules out FH-IPFE schemes with correctness probability less than $1/2$, which strengthens our impossibility result also in Maurer's model.

3 Preliminaries

Notation. We use κ to denote the security parameter. We use $[n] := \{1, \ldots, n\}$ for $n \in \mathbb{N}$. For two functions $f, g \colon \mathbb{N} \to \mathbb{R}$, when we say $f \geqslant g$, we mean that this holds asymptotically; i.e., there exists $c \in \mathbb{N}$ such that for all $x \geqslant c$, $f(x) \geqslant g(x)$. For a discrete distribution D, we use $\mathsf{D}(a)$ to mean the output of the probability mass function of D on point a; i.e., $\mathsf{D}(a) = \Pr_{x \leftarrow \mathsf{D}}[x = a]$. We write $x \leftarrow \mathsf{D}$ to denote sampling x from D and write $x \in \mathsf{D}$ to indicate x is in the support of D.

We use $\mathsf{Span}(\mathsf{S})$ to denote the linear span of a set of vectors $\mathsf{S} \subseteq \mathbb{Z}_q^n$ or the column vectors of a matrix $\mathsf{S} \in \mathbb{Z}_q^{n \times m}$. For a $\mathbb{Z}_q$ module $\mathsf{V} \subseteq \mathbb{Z}_q^n$ we write $\mathsf{V}^\perp := \{\mathbf{w} \in \mathbb{Z}_q^n \mid \forall \mathbf{v} \in \mathsf{V} : \langle \mathbf{w}, \mathbf{v} \rangle = 0\}$ and similarly for a vector $\mathbf{x} \in \mathbb{Z}_q^n$ we write $\mathbf{x}^\perp := \{\mathbf{w} \in \mathbb{Z}_q^n \mid \langle \mathbf{w}, \mathbf{x} \rangle = 0\}$.

Lemma 3.1 (Chernoff bound). *Let $X_1, \ldots, X_m$ be i.i.d. Bernoulli random variables where $\mu = \Pr[X_1 = 1]$. Let $X = \frac{X_1+\cdots+X_m}{m}$. Then*

$$\Pr[|X - \mu| \geqslant \epsilon] \leqslant 2 \cdot e^{-\frac{m\epsilon^2}{4}}.$$

3.1 Inner-Product Functional Encryption

We define private-key IPFE schemes relative to an oracle. Later on, we define a GGM-based definition of IPFE to capture the scenario where the decryption algorithm learns the inner product in the exponent (Definition 3.5).

Definition 3.2 (IPFE). A private-key IPFE scheme $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$ relative to an oracle O for $\mathbb{Z}_q^n$ with dimension $n = n(\kappa)$ and modulus $q = q(\kappa)$ is given by the following oracle-aided PPT algorithms.

- $\mathrm{Setup}^O(1^\kappa)$: On input a unary encoded security parameter, it outputs a master secret key msk.
- $\mathrm{KGen}^O(\mathsf{msk}, \mathbf{y})$: On input a master secret key msk and a vector $\mathbf{y} \in \mathbb{Z}_q^n$, the key generation algorithm outputs a secret key $\mathsf{sk}[\mathbf{y}]$.
- $\mathrm{Enc}^O(\mathsf{msk}, \mathbf{x})$: On input a master secret key msk and a vector $\mathbf{x} \in \mathbb{Z}_q^n$, the encryption algorithm outputs a ciphertext ct.
- $\mathrm{Dec}^O(\mathsf{sk}[\mathbf{y}], \mathsf{ct})$: On input a secret key $\mathsf{sk}[\mathbf{y}]$ and a ciphertext ct, the decryption algorithm outputs $z \in \mathbb{Z}_q$.

- **ν-correctness**: We say $\mathcal{E}^O$ is ν-correct if the following experiment outputs one with probability at least ν. Sample $\mathsf{msk} \leftarrow \mathrm{Setup}^O(1^\kappa)$, $\mathbf{y} \leftarrow \mathbb{Z}_q^n$, $\mathsf{sk}[\mathbf{y}] \leftarrow \mathrm{KGen}^O(\mathsf{msk}, \mathbf{y})$, $\mathbf{x} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{ct} \leftarrow \mathrm{Enc}^O(\mathsf{msk}, \mathbf{x})$. The experiment outputs one if $\mathrm{Dec}^O(\mathsf{sk}[\mathbf{y}, \mathsf{ct}) = \langle \mathbf{y}, \mathbf{x} \rangle$. If the oracle O is randomized, the probability is taken also over the random coins of the oracle.

Definitions for the impossibility result. We introduce two new security notions that are parameterized by a fixed $(n-1)$-dimensional subspace V. In Appendix D we show that the standard FH-IND-CPA security notion implies both of these notions for any V.

Definition 3.3 (V-key hiding). Let $\mathsf{V} \subseteq \mathbb{Z}_q^n$ be an $(n-1)$-dimensional subspace of $\mathbb{Z}_q^n$ and $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$ be an IPFE over $\mathbb{Z}_q^n$. The game is defined over a PPT adversary $\mathcal{A}$ and a challenger. The challenger samples a master secret key $\mathsf{msk} \leftarrow \mathrm{Setup}^O(1^\kappa)$ and a challenge bit $b \leftarrow \{0, 1\}$. The adversary $\mathcal{A}$, who has the description of V, can make queries of the following types, where neither type requires an input from $\mathcal{A}$, and where all the queries must be made non-adaptively at once.

- **Key corruption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{y}, \mathbf{y} + \mathbf{w}, \mathrm{KGen}(\mathsf{msk}, \mathbf{y} + b\mathbf{w}))$, where $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{w} \leftarrow \mathsf{V}^\perp$. Here $\mathsf{V}^\perp$ denotes the orthogonal complement of V; i.e., $\mathsf{V}^\perp = \{\mathbf{w} \in \mathbb{Z}_q^n \mid \forall \mathbf{v} \in \mathsf{V} : \langle \mathbf{w}, \mathbf{v} \rangle = 0\}$.
- **Encryption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{v}, \mathrm{Enc}(\mathsf{msk}, \mathbf{v}))$, where $\mathbf{v} \leftarrow \mathsf{V}$.

At the end, $\mathcal{A}$ outputs a bit b' and wins if $b = b'$. We say $\mathcal{A}$ is admissible if all of $\mathcal{A}$'s queries are made non-adaptively at once. We say $\mathcal{E}$ is V-key hiding if any admissible PPT $\mathcal{A}$ wins the game with probability at most $1/2 + \mathrm{negl}(\kappa)$.

Note that the above definition can be realized under the DDH assumption, since it only concerns key hiding, and not plaintext hiding. In particular, the dual version of any IPFE that satisfies the standard notion of IND-CPA security satisfies the above notion (e.g., [2]). Dual means swapping the roles of the key-generation and encryption algorithms. Thus, the above definition, in isolation, cannot be ruled out in the GGM. Thus, we give a definition for plaintext hiding below, and will show that these two definitions cannot be simultaneously realized in the GGM.

Definition 3.4 (V-plaintext hiding). Let $\mathsf{V} \subseteq \mathbb{Z}_q^n$ be an $(n-1)$-dimensional subspace of $\mathbb{Z}_q^n$ and $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$ be an IPFE over $\mathbb{Z}_q^n$. Let msk and bit b be sampled as in Definition 3.3. The adversary $\mathcal{A}$ may make queries of the following types, all made non-adaptively at once.

- **Key corruption queries**: The adversary $\mathcal{A}$ submits $\mathbf{y}$ subject to the restriction that $\mathbf{y} \notin \mathsf{V}^\perp$ and receives $\mathrm{KGen}(\mathsf{msk}, \mathbf{y})$. The adversary may make this query only once.
- **LoR encryption queries**: For any such query, $\mathcal{A}$ receives $(\langle \mathbf{y}, \mathbf{x} \rangle, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}))$, where $\mathbf{x} = \mathbf{v} + b\mathbf{x}'$ for $\mathbf{v} \leftarrow \mathsf{V}$ and $\mathbf{x}' \leftarrow \mathbb{Z}_q^n$.
- **Straight encryption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{x}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}))$, where $\mathbf{x} \leftarrow \mathbb{Z}_q^n$.

At the end, $\mathcal{A}$ outputs a bit b' and wins if $b = b'$. We say that $\mathcal{A}$ is admissible if $\mathcal{A}$ makes at most one key corruption query and if all of $\mathcal{A}$'s queries are made non-adaptively at once. We say that $\mathcal{E}$ is V-plaintext hiding if any admissible PPT $\mathcal{A}$ wins the game with probability at most $1/2 + \mathrm{negl}(\kappa)$.

We now give an adaptation of the IPFE notion of Definition 3.2 to the GGM setting, capturing the common scenario where the decryption algorithm learns the inner product only in the exponent. Here $\mathbb{G}$ is the generic group oracle (in Maurer's or Shoup's variant) that we describe in more detail in Appendices A and B.

Definition 3.5 (IPFE in GGM). The algorithms $\mathrm{Setup}^{\mathbb{G}}$, $\mathrm{KGen}^{\mathbb{G}}$ and $\mathrm{Enc}^{\mathbb{G}}$ are the same as in Definition 3.2. We define $\mathrm{Dec}^{\mathbb{G}}$ as follows: it takes in $(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], z)$, where z is a candidate inner product value, and $\mathrm{Dec}^{\mathbb{G}}$ outputs $\{0,1\}$ while having the following structure: $\mathrm{Dec}^{\mathbb{G}} = (\mathrm{Dec}_0^{\mathbb{G}}, \mathrm{Dec}_1^{\mathbb{G}})$, where $\mathrm{Dec}_0^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}])$ outputs a group element $\widetilde{g}$, and $\mathrm{Dec}_1^{\mathbb{G}}$ outputs the result of an equality query on $(\widetilde{g}, zg)$, where g is the generator.[5] The output of Dec is the output of Dec_1. Also, note that Dec_0 does not take in z. For ν-correctness we require for $\mathsf{msk} \leftarrow \mathrm{Setup}^{\mathbb{G}}(1^\kappa)$, for $\mathbf{y}, \mathbf{x} \leftarrow \mathbb{Z}_q^n$, $\mathsf{sk}[\mathbf{y}] \leftarrow \mathrm{KGen}^{\mathbb{G}}(\mathsf{msk}, \mathbf{y})$ and $\mathsf{ct}[\mathbf{x}] \leftarrow \mathrm{Enc}^{\mathbb{G}}(\mathsf{msk}, \mathbf{x})$, the probability that $\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], \langle \mathbf{y}, \mathbf{x} \rangle)$ outputs 1 is at least ν.

4 Impossibility of FH-IPFE in Maurer's GGM

In this section, we present our impossibility result in Maurer's GGM. We refer the reader to Definition A.1 for a definition of Maurer's GGM.

For simplicity, we assume for our impossibility results that q is a prime and thus $\mathbb{Z}_q^n$ is a vector space and not just a module. This is without loss of generality, because an IPFE for a composite modulus q can be transformed into an IPFE for any prime modulus $p \mid q$. The transformation simply multiplies in key generation the input vector with q/p and the decryption algorithm divides the result by q/p.

[5] Technically, this means making a sequence of Add queries to compute xg from g and comparing it against $\widetilde{g}$.

4.1 Step 1: Removing oracle queries from KGen

Lemma 4.1 *Assume* $(\mathrm{Setup}^{\mathbb{G}}, \mathrm{KGen}^{\mathbb{G}}, \mathrm{Enc}^{\mathbb{G}}, \mathrm{Dec}^{\mathbb{G}})$ *is* ν*-correct and satisfies both* V*-key hiding (Definition 3.3) and* V*-plaintext hiding (Definition 3.4) for every* $(n-1)$*-dimensional subspace* V*. The compiled scheme* $\overline{\mathcal{E}}^{\mathbb{G}} := (\overline{\mathrm{Setup}}^{\mathbb{G}}, \overline{\mathrm{KGen}}, \overline{\mathrm{Enc}}^{\mathbb{G}}, \overline{\mathrm{Dec}}^{\mathbb{G}})$*, defined for any given* V*, has the property that* $\overline{\mathrm{KGen}}$ *makes no* $\mathbb{G}$ *queries, and* $\overline{\mathrm{Setup}}^{\mathbb{G}}, \overline{\mathrm{KGen}}, \overline{\mathrm{Enc}}^{\mathbb{G}}, \overline{\mathrm{Dec}}^{\mathbb{G}}$ *continue to be PPT algorithms, and*

1. $\overline{\mathcal{E}}^{\mathbb{G}}$ *is* $((3\nu-2) - 1/q^{n-1} - 1/\mathsf{poly}(\kappa))$*-correct for an arbitrarily chosen polynomial* $\mathsf{poly}(\kappa)$ *and for some subspace* V*.*[6]
2. $\overline{\mathcal{E}}^{\mathbb{G}}$ *provides* V*-key hiding (Definition 3.3) for any subspace* V*.*

Proof of Lemma 4.1. Our compiler is presented in Figure 2. It is parametrized by an $(n-1)$-dimensional subspace $\mathsf{V} \subseteq \mathbb{Z}_q^n$, and compiles $(\mathrm{Setup}^{\mathbb{G}}, \mathrm{KGen}^{\mathbb{G}}, \mathrm{Enc}^{\mathbb{G}}, \mathrm{Dec}^{\mathbb{G}})$ into $\overline{\mathcal{E}}^{\mathbb{G}} := (\overline{\mathrm{Setup}}^{\mathbb{G}}, \overline{\mathrm{KGen}}, \overline{\mathrm{Enc}}^{\mathbb{G}}, \overline{\mathrm{Dec}}^{\mathbb{G}})$.

Remark 4.2 For a GGM algorithm $A^{\mathbb{G}}(g_1, \ldots, g_m)$ with initial input group elements $g_1, \ldots, g_m$, we define a notion of positively-answered equality relations. In particular, for any equality query $\mathsf{Eql}(g', g'')$ answered with 1, the algorithm A may map each g' and g'' to a linear combination over $(g_1, \ldots, g_m)$, hence obtaining a satisfying relation $(a_1, \ldots, a_m)$ over $(g_1, \ldots, g_m)$; i.e., $a_1 g_1 + \cdots + a_m g_m = 0$.

$\overline{\mathrm{KGen}}$ makes no $\mathbb{G}$ queries. We note that $\overline{\mathrm{KGen}}$ indeed does not require making any $\mathbb{G}$ queries. Firstly, $\overline{\mathrm{KGen}}$ does not need to output any group elements beyond pp, which is already prepared in $\overline{\mathsf{msk}}$ by $\overline{\mathrm{Setup}}^{\mathbb{G}}$. Secondly, for a given Add query, $\overline{\mathrm{KGen}}$ may create a 'dummy' entry in the table and give a pointer to that dummy cell. This creates no issues because $\overline{\mathrm{KGen}}$ is not supposed to output any of such dummy entries (only those in pp are output). Finally, for an equality query asked during the invocation of $\mathrm{KGen}^{\mathbb{G}}$ or during the preparation of R, $\overline{\mathrm{KGen}}$ may simulate the response itself. $\overline{\mathrm{KGen}}$ is capable of doing this because it takes $\overline{\mathsf{msk}}$ as input and, hence, it knows the discrete logs (i.e., $T(i)$) for all the group elements created during $\overline{\mathrm{Setup}}$ (given in aux) and also during $\overline{\mathrm{KGen}}$'s execution so far. Consequently, $\overline{\mathrm{KGen}}$ can answer all the equality queries by examining the underlying discrete log values.

V-key hiding. Next, we argue the compiled scheme satisfies V-key hiding. We do so by showing that, if there is an adversary $\mathcal{A}$ that breaks the V-key hiding property, we can build another adversary $\mathcal{A}^*$ that breaks the V-key hiding property of the base scheme.

In particular, $\mathcal{A}^*$ will first make ℓ encryption queries and obtain the $\mathsf{pp} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ in the compiled scheme, together with the underlying plaintexts $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\ell$. Then, $\mathcal{A}^*$ proceeds to simulate the view of $\mathcal{A}$ for the key hiding game for the compiled scheme. For each encryption query, $\mathcal{A}^*$ simply relays the query to $\mathcal{A}^*$'s own external challenger and responds to $\mathcal{A}$ whatever it receives from the challenger. For each key corruption query, $\mathcal{A}^*$ will first relay this query to its own external challenger to receive sk. Next, it proceeds to augment its received sk in the same way the compiler does. Note that $\mathcal{A}^*$ is able to do so since it holds pp which contains $(\mathbf{v}_i, \mathsf{ct}_i)$ for all $i \in [\ell]$ — which, in turn, means that $\mathcal{A}^*$ knows the inner product used as input to Dec. $\mathcal{A}^*$ then gives the compiled key to $\mathcal{A}$.

Clearly, $\mathcal{A}^*$ perfectly simulates the view of $\mathcal{A}$ and the V-key hiding of the original scheme implies the V-key hiding of the compiled scheme. □

[6] Note that q represents the modulus of the IPFE, which does not necessarily agree with the order of the group p.

$\overline{\mathsf{Setup}}^{\mathbb{G}}(1^\kappa)$: Sample msk according to $\mathsf{Setup}^{\mathbb{G}}(1^\kappa)$. Let ℓ be an integer which will be instantiated later (in the proof for correctness). For $i \in [\ell]$, sample $\mathbf{v}_i \leftarrow \mathsf{V}$ and $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathbb{G}}(\mathsf{msk}, \mathbf{v}_i)$. Let the resulting master secret key $\overline{\mathsf{msk}}$ contain msk and $\mathsf{pp} := (\mathbf{v}_1, \mathsf{ct}_1, \ldots, \mathbf{v}_\ell, \mathsf{ct}_\ell)$, as well as aux, which contains all the randomness and the underlying query-answer (Q-A) pairs (i.e., all the discrete log values) made throughout.
For $i \in [\ell]$, let $(g_{i,1}, \ldots, g_{i,t})$ denote the group elements in ct_i.

$\overline{\mathsf{KGen}}(\overline{\mathsf{msk}} := (\mathsf{msk}, \mathsf{pp}, \mathsf{aux}), \mathbf{y})$**:** Sample $\mathsf{sk}[\mathbf{y}] := (s, g_1, \ldots, g_m) \leftarrow \mathsf{KGen}^{\mathbb{G}}(\mathsf{msk}, \mathbf{y})$, where s is a bit string and g_i's are group elements. The oracle $\mathbb{G}$ is simulated using aux without making any queries to the real oracle. Let the compiled key be $\overline{\mathsf{sk}[\mathbf{y}]} := (s, \mathsf{pp}, \mathsf{R})$, where R is a set of relations over $(g_1, \ldots, g_m, \{g_{i,j}\}_{i\in[\ell], j\in[t]})$, formed as follows.

1. For all $h \in [\ell]$, perform $\mathsf{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}_h, \langle \mathbf{y}, \mathbf{v}_h \rangle)$. For any positively-answered equality query $\sum_{i=1}^m a_i \cdot g_i + \sum_{i=1}^t a'_{h,i} \cdot g_{h,i} = 0$, add the corresponding coefficients (i.e., $(a_1, \ldots, a_m,\ a'_{h,1}, \ldots, a'_{h,t})$) to R (Remark 4.2).

$\overline{\mathsf{Enc}}^{\mathbb{G}}(\overline{\mathsf{msk}} := (\mathsf{msk}, \mathsf{pp}, \mathsf{aux}), \mathbf{x})$**:** output $\mathsf{Enc}^{\mathbb{G}}(\mathsf{msk}, \mathbf{x})$.

$\overline{\mathsf{Dec}}^{\mathbb{G}}\left(\overline{\mathsf{sk}[\mathbf{y}]} := (s, \mathsf{pp}, \mathsf{R}), \mathsf{ct} := (s', u_1, \ldots, u_t), z\right)$**:** To decrypt a ciphertext $\mathsf{ct} := (s', u_1, \ldots, u_t)$, where s' is a bit string and u_i's are the group elements, perform $\mathsf{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}, z)$. The decryptor can perform all operations except for equality queries (since the secret key no longer contains the group elements $(g_1, \ldots, g_m)$, but only contains the relations R). Instead, $\overline{\mathsf{Dec}}^{\mathbb{G}}$ answers such equality queries as follows.

(a) Any equality query must be of the form $\sum_{i=1}^m z_i \cdot g_i + \sum_{i=1}^t c_i \cdot u_i = 0$. Now, if $(z_1, \ldots, z_m) \notin \mathsf{Span}((\mathsf{R}')^\top)$, where R' is the submatrix of R that contains the columns that correspond to $(g_1, \ldots, g_m)$, answer the equality query with 0. Otherwise, cancel out g_i's using the relations in R and obtain an equivalent equality query that involves instead $\{g_{i,j}\}_{i\in[\ell], j\in[t]}$ and $\{u_i\}_{i\in[t]}$. We can ask $\mathbb{G}$ on this relation.

We remind the reader that, without loss of generality, we may assume that $\mathsf{Dec}^{\mathbb{G}}$ does not make any Add queries (Remark A.2) and, hence, we only need to worry about equality queries.

Figure 2: The compiler for Lemma 4.1, parameterized by a subspace $\mathsf{V} \subseteq \mathbb{Z}_q^n$.

Correctness. Next, we argue correctness. When we invoke the our compiler using a subspace V, define the correctness specific to V as

$$\mathsf{Correct}_{\mathsf{V}} := \Pr\left[\overline{\mathsf{Dec}}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}, \langle \mathbf{y}, \mathbf{x} \rangle) = 1 \;\middle|\; \begin{array}{c} \overline{\mathsf{msk}} \leftarrow \overline{\mathsf{Setup}}^{\mathbb{G}}(1^\kappa) \\ \mathbf{y} \leftarrow \mathbb{Z}_q^n,\ \mathbf{x} \leftarrow \mathbb{Z}_q^n \\ \overline{\mathsf{sk}[\mathbf{y}]} \leftarrow \overline{\mathsf{KGen}}(\overline{\mathsf{msk}}, \mathbf{y}) \\ \mathsf{ct} \leftarrow \overline{\mathsf{Enc}}^{\mathbb{G}}(\overline{\mathsf{msk}}, \mathbf{x}) \end{array}\right].$$

We are going to prove that

$$\mathop{\mathrm{E}}_{\mathsf{V}}[\mathsf{Correct}_{\mathsf{V}}] \geqslant (3\nu - 2) - 1/q^{n-1} - 3m/(\ell + 1) - \mathrm{negl}(\kappa). \tag{3}$$

This suffices to prove Lemma 4.1 since Equation 3 states that the correctness of the compile scheme with respect to a random parameter V is sufficiently high, which implies the existence of a subspace V such that the compilation using this particular V satisfies the above bound. Henceforth, the requirement of Lemma 4.1 is met by setting ℓ to be a sufficiently large polynomial.[7]

For this proof, let us consider a slightly different compiled scheme, where the only difference lies in the choice of V (within $\overline{\mathsf{Setup}}^{\mathbb{G}}$) for the compiler. Instead of picking V as a subspace of dimension $n-1$, we pick V to be the *entire space*. Our proof breaks into the following two steps.

1. For this slightly different compiled scheme, where V is the entire space, its correctness $\geqslant \nu - \frac{m}{\ell+1}$.
2. When we switch V from the entire space to a random dimension-$(n-1)$ subspace, the correctness will not drop significantly.

Proof of Step 1. For this step, let us first import a result from [38].

Lemma 4.3 (Lemma 3.2 of [38]). *Let $X_1, X_2, \ldots, X_k$ be i.i.d. random variables over subsets of a linear vector space V of dimension $\mathsf{dim}(\mathsf{V})$. Then*

$$\Pr[X_k \not\subseteq \mathsf{span}(X_1 \cup X_2 \cup \cdots \cup X_{k-1})] \leqslant \frac{\mathsf{dim}(\mathsf{V})}{k}.$$

By the correctness of the base scheme,

$$\Pr\left[\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}, \langle \mathbf{y}, \mathbf{x}\rangle) = 1 \;\middle|\; \begin{array}{c} \mathsf{msk} \leftarrow \mathrm{Setup}^{\mathbb{G}}(1^\kappa) \\ \mathbf{y} \leftarrow \mathbb{Z}_q^n,\ \mathbf{x} \leftarrow \mathbb{Z}_q^n \\ \mathsf{sk}[\mathbf{y}] \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{y}) \\ \mathsf{ct} \leftarrow \mathrm{Enc}^{\mathbb{G}}(\mathsf{msk}, \mathbf{x}) \end{array}\right] \geqslant \nu. \tag{4}$$

The compiled scheme has the same encryption process and "almost" the same decryption process except for when an equality query is issued. For $\mathrm{Dec}^{\mathbb{G}}$, such equality queries are answered by the GGM oracle. For $\overline{\mathrm{Dec}}^{\mathbb{G}}$, such equality queries are answered by the linear relation R made by the compiler.

For Equations 3 and 4 let $\mathsf{sk}[\mathbf{y}] = (s, g_1, \ldots, g_m)$ be the base secret key and $\overline{\mathsf{sk}[\mathbf{y}]} := (s, \mathsf{pp}, R)$ be the compiled secret key. The only time that the random process in Equation 3 and Equation 4 might produce different outputs is when equality queries asked during the two decryption processes result in different outputs. That is,

$$\overline{\mathrm{Dec}}^{\mathbb{G}}(\overline{\mathsf{sk}[\mathbf{y}]}, \mathsf{ct}, \langle \mathbf{y}, \mathbf{x}\rangle) \neq \mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}, \langle \mathbf{y}, \mathbf{x}\rangle)$$

only when some equality query is answered differently between $\overline{\mathrm{Dec}}^{\mathbb{G}}$ and $\mathrm{Dec}^{\mathbb{G}}$. We are going to argue that this can only happen with probability at most $\frac{m}{\ell+1}$, which would conclude the proof of Step 1.

We define the following variables.

- X_h for $h \in [\ell]$: for any positively-answered relation $\sum_{i=1}^m a_i \cdot g_i + \sum_{i=1}^t a'_{h,i} \cdot g_{h,i} = 0$ during the execution of $\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}_h)$ performed by $\overline{\mathrm{KGen}}$ (Step 1 of Figure 2), add the vector $(a_1, \ldots, a_m)$ to X_h, the corresponding coefficients underlying $(g_1, \ldots, g_m)$.

[7] Note that all the other requirements of Lemma 4.1 hold for every possible V. Correctness is the only requirement where we argue it for a random subspace V.

- $X_{\ell+1}$: Let $u_1, \ldots, u_t$ be the group elements in the ciphertext ct (Equations 3 and 4). For any positively-answered relation $\sum_{i=1}^{m} a_i \cdot g_i + \sum_{i=1}^{t} a'_i \cdot u_i = 0$ during the execution of $\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct})$, add the vector $(a_1, \ldots, a_m)$ to $X_{\ell+1}$, the corresponding coefficients underlying $(g_1, \ldots, g_m)$.

Let Good be the event that $X_{\ell+1} \subseteq \mathsf{span}(X_1 \cup X_2 \cup \cdots \cup X_\ell)$. By Lemma 4.3, $\Pr[\overline{\mathsf{Good}}] \leqslant \frac{m}{\ell+1}$. Moreover, we claim (*) that whenever Good holds, then the decryption outputs will be consistent, namely $\overline{\mathrm{Dec}}^{\mathbb{G}}(\overline{\mathsf{sk}[\mathbf{y}]}, \mathsf{ct}, \langle \mathbf{y}, \mathbf{x} \rangle) = \mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}, \langle \mathbf{y}, \mathbf{x} \rangle)$. This will conclude the proof of Step 1. Claim (*) follows because as we will show if Good holds, $\overline{\mathrm{Dec}}^{\mathbb{G}}$ will simulate correctly the response to any equality query of the form

$$\sum_{i=1}^{m} z_i \cdot g_i + \sum_{i=1}^{t} c_i \cdot u_i = 0. \tag{5}$$

Note that $\overline{\mathrm{Dec}}^{\mathbb{G}}$ does not have explicit access to the group elements $(g_1, \ldots, g_m)$, which are those in the secret key $\mathsf{sk}[\mathbf{y}]$, and should make up for its lack of access via the prepared matrix R. Let R' be the submatrix of R that contains only the columns that correspond to $(g_1, \ldots, g_m)$ (Step a of Figure 2). We consider all possible cases.

- $(z_1, \ldots, z_m) \in \mathsf{Span}(X_1 \cup \cdots \cup X_\ell)$: In this case $\overline{\mathrm{Dec}}$ reduces the relation to an equivalent relation involving group elements $\{u_i\}$ and $\{g_{i,j}\}$ that it has explicit access to, and so $\overline{\mathrm{Dec}}$ will return a consistent response (Step a of Figure 2).
- $(z_1, \ldots, z_m) \notin \mathsf{Span}(X_1 \cup \cdots \cup X_\ell)$ and the relation of Equation 5 does not hold: In this case both Dec and $\overline{\mathrm{Dec}}$ respond to the query with 0.
- $(z_1, \ldots, z_m) \notin \mathsf{Span}(X_1 \cup \cdots \cup X_\ell)$ and the relation of Equation 5 holds: This case cannot happen because we assumed Good holds.

The proof of Step 1 is now complete. □

Proof of Step 2. In this step, we argue that the correctness does not drop by too much when we switch from the entire space to a random $(n-1)$-dimensional subspace. For ease of presentation, let us introduce some notations. Let Corr be the following randomized process.

$$\mathsf{Corr} := \left\{ \begin{array}{c} \mathsf{msk} \leftarrow \mathrm{Setup}^{\mathbb{G}}(1^\kappa) \\ \mathbf{y} \leftarrow \mathbb{Z}_q^n, \ \mathbf{x} \leftarrow \mathbb{Z}_q^n \\ \mathsf{sk}[\mathbf{y}] \leftarrow \mathrm{KGen}^{\mathbb{G}}(\mathsf{msk}, \mathbf{y}) \\ \mathsf{ct} \leftarrow \mathrm{Enc}^{\mathbb{G}}(\mathsf{msk}, \mathbf{x}) \end{array} \right\}.$$

Note that Corr is defined using the base IPFE scheme. In the rest of the analysis, all the probabilities we consider are computed over the sampling of Corr. Let $\mathrm{SubDec}_{\mathsf{V}}$ denote the following process.

1. Sample ℓ ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ encrypting random vectors from V, using msk of Corr.
2. Compile $\mathsf{sk}[\mathbf{y}]$, given in Corr, by learning the relation R just as in the compiler of Figure 2 (Step 1). To do this step, we make use of the oracle $\mathbb{G}$. Let the compiled secret key be $\overline{\mathsf{sk}[\mathbf{y}]}$.
3. Run $\overline{\mathrm{Dec}}^{\mathbb{G}}(\overline{\mathsf{sk}[\mathbf{y}]}, \mathsf{ct}, \langle \mathbf{y}, \mathbf{x} \rangle)$, where recall that ct was defined in Corr.

Let FullDec denote the similar process except V is the entire space. Under this notation, what we have proved in Step 1 is

$$\Pr[\mathrm{FullDec} = 1] \geqslant \nu - \frac{m}{\ell+1}.$$

We need to argue that $\Pr[\mathrm{SubDec}_{\mathsf{V}} \neq 1]$ is small. Consider the following derivation.

$$\begin{aligned}
&\Pr[\mathrm{SubDec}_{\mathsf{V}} \neq 1] \\
&\leqslant \Pr[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec}] + \Pr[\mathrm{FullDec} \neq 1] \\
&\leqslant \Pr[\mathbf{y} \in \mathsf{V}^{\perp}] + \Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big] + \left(1 - \nu + \frac{m}{\ell+1}\right) \\
&\leqslant \frac{1}{q^{n-1}} + \Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big] + \left(1 - \nu + \frac{m}{\ell+1}\right).
\end{aligned}$$

We complete the proof of correctness by proving that, due to V-plaintext hiding (Definition 3.4) property, we have

$$\mathop{\mathrm{E}}_{\mathsf{V}}\Big[\Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big]\Big] \leqslant 2 \cdot \left(1 - \nu + \frac{m}{\ell+1}\right) + \mathrm{negl}(\kappa).$$

This suffices because

$$\begin{aligned}
&\mathop{\mathrm{E}}_{\mathsf{V}}[\Pr[\mathrm{SubDec}_{\mathsf{V}} \neq 1]] \\
&\leqslant \mathop{\mathrm{E}}_{\mathsf{V}}\left[\frac{1}{q^{n-1}} + \Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big] + \left(1 - \nu + \frac{m}{\ell+1}\right)\right] \\
&\leqslant \frac{1}{q^{n-1}} + \mathop{\mathrm{E}}_{\mathsf{V}}\Big[\Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big]\Big] + \left(1 - \nu + \frac{m}{\ell+1}\right) \\
&\leqslant \frac{1}{q^{n-1}} + 2 \cdot \left(1 - \nu + \frac{m}{\ell+1}\right) + \mathrm{negl}(\kappa) + \left(1 - \nu + \frac{m}{\ell+1}\right),
\end{aligned}$$

which proves Equation 3 and, henceforth, the correctness guarantee.

Let us run the V-plaintext hiding game, where the adversary makes one key corruption query (on the vector $\mathbf{y}$) to receive $\mathsf{sk}[\mathbf{y}]$, and makes both (a) ℓ many LoR queries and (b) ℓ many straight encryption queries. Now we can run the compiler twice on $\mathsf{sk}[\mathbf{y}]$: (a) once using the ℓ ciphertexts from the LoR queries and (b) once using the ℓ ciphertexts from the straight encryption queries. Note that we can run the compiler since we always know *the inner product between the secret key vector* $\mathbf{y}$ *and the plaintext vector that the ciphertext encrypts* (refer to Definition 3.4 that we can either compute this inner product ourselves or receive it from the oracle). This information is used in the compiler to run decryption $\mathrm{Dec}^{\mathbb{G}}$.

Finally, we ask for one more random ciphertext ct from the straight encryption query and try to decrypt ct using the two compiled secret keys to see if they result in the same decryption or not. When the bit $b = 1$, this corresponds to the probability

$$\Pr\Big[\mathrm{FullDec} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big], \tag{6}$$

where the two FullDec are two independent runs of FullDec with fresh randomness (but under the same $\mathsf{sk}[\mathbf{y}]$ and ct). On the other hand, when the bit $b = 0$, the probability is exactly

$$\Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^{\perp}\Big]. \tag{7}$$

By the plaintext hiding property, we know that the two probabilities in Equation 6 and Equation 7 should only differ by $\mathrm{negl}(\kappa)$. Finally, when we run FullDec twice, the probability that these two

decryptions are different is upper-bounded by twice the correctness error (i.e., $2 \cdot (1 - \nu + \frac{m}{\ell+1})$). Note that the conditioning on $\mathbf{y} \notin \mathsf{V}^\perp$ does not change the correctness error when we average over the choice of V because V is a random subspace independent of all other randomness. That is,

$$\mathop{\mathrm{E}}_{\mathsf{V}}\Big[\Pr\Big[\mathrm{FullDec} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^\perp\Big]\Big] \leqslant 2 \cdot \left(1 - \nu + \frac{m}{\ell+1}\right)$$

This concludes the proof since we now obtain

$$\mathop{\mathrm{E}}_{\mathsf{V}}\Big[\Pr\Big[\mathrm{SubDec}_{\mathsf{V}} \neq \mathrm{FullDec} \;\Big|\; \mathbf{y} \notin \mathsf{V}^\perp\Big]\Big] \leqslant 2 \cdot \left(1 - \nu + \frac{m}{\ell+1}\right) + \mathrm{negl}(\kappa).$$

4.2 Step 2: Breaking Key Hiding for Oracle-Free KGen

In this section we show how to break the V-key hiding of any given IPFE, for which KGen makes no oracle queries. Our attack is general and applies to any oracle O. The described attacker does not make any queries to O at all, and in fact breaks even *weak* V-key-hiding: defined as in Definition 3.3, where the adversary is not allowed to make any encryption queries, and can only make key-corruption queries. Also, we give the attack for the general IPFE schemes of Definition 3.2. We explain in Section A.1 the changes we need to make to the proof to adapt it to the two-phase decryption algorithms as in Definition 3.5.

Lemma 4.4 *Let $\mathcal{E}^O := (\mathrm{Setup}^O, \mathrm{KGen}, \mathrm{Enc}^O, \mathrm{Dec}^O)$ be an IPFE over $\mathbb{Z}_q^n$ with correctness of at least 0.999, where* KGen *makes no oracle queries. Suppose $n \geqslant 2$. Then, there exists an attacker $\mathcal{A}$ that makes no O queries and wins the weak V-key hiding game, for any given V, with probability $1/2 + \frac{1}{\mathsf{poly}(\kappa)}$ for some polynomial poly.*

Proof. We give the description of the attacker. Suppose $|\mathsf{msk}| = \rho$. Let $m = 16(\kappa + \rho)\kappa^2$. The adversary $\mathcal{A}$ makes m key-corruption queries to receive $(\mathbf{w}_1, \mathbf{u_1} := \mathbf{w}_1 + \mathbf{v}_1, \mathsf{sk}_1), \ldots$ $(\mathbf{w}_m, \mathbf{u}_m := \mathbf{w}_m + \mathbf{v}_m, \mathsf{sk}_m)$. Recall that for all $i \in [m]$: $\mathbf{w}_i \leftarrow \mathbb{Z}_q^n$, $\mathbf{v}_i \leftarrow \mathsf{V}^\perp$. The adversary attempts decoding the m given secret keys via a procedure Recover, defined momentarily, to obtain m recovered vectors, one for each coordinate $i \in [m]$. Specifically, $\mathcal{A}$ works as follows.

1. If $\exists \widetilde{\mathsf{msk}} \in \{0,1\}^\rho$ such that for at least $m(\frac{1}{q} + \frac{1}{2\kappa})$ coordinates i, $\mathbf{w}_i = \mathbf{h}_i$, where $(\mathbf{h}_1, \ldots, \mathbf{h}_m) := \mathrm{Recover}(\widetilde{\mathsf{msk}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$, return $b' = 0$. Else, return $b' = 1$.

The procedure Recover recovers the ith vector by examining the distribution over vectors obtained by reverse sampling a vector based on $(\widetilde{\mathsf{msk}}, \mathsf{sk}_i)$. Then, it outputs the vector with weight strictly greater than $1/2$ (if any). Details follow.

Procedure $\mathrm{Recover}(\widetilde{\mathsf{msk}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$**:**

1. For $i \in [m]$, form the vector $\mathbf{h}_i$ as follows.
   (a) Let D_i be the induced distribution over uniformly-random vectors conditioned on $(\widetilde{\mathsf{msk}}, \mathsf{sk}_i)$:
   $$\mathsf{D}_i(\mathbf{v}) = \Pr_{\mathbf{x} \leftarrow \mathbb{Z}_q^n}[\mathbf{x} = \mathbf{v} \mid \mathsf{sk}_i = \mathrm{KGen}(\widetilde{\mathsf{msk}}, \mathbf{x})]. \tag{8}$$
   (b) If there exists a vector $\mathbf{v}$ such that $\mathsf{D}_i(\mathbf{v}) > \frac{1}{2}$, set $\mathbf{h}_i = \mathbf{v}$. Else, set $\mathbf{h}_i = \perp$.
2. Return $(\mathbf{h}_1, \ldots, \mathbf{h}_m)$.

First, note that the above computation, especially that of Recover, involves no oracle queries. This is where we make use of the fact that KGen makes no oracle calls. Moreover, $\mathcal{A}$ makes no encryption queries in the key-hiding game (Definition 3.3), so $\mathcal{A}$ indeed breaks *weak* V-key hiding.

The proof of Lemma 4.4 follows immediately from the following two lemmas. In the following b denotes the challenge bit. □

Lemma 4.5 *We have* $\Pr[b' = 0 \mid b = 0] \geqslant \frac{1}{\mathsf{poly}(\kappa)}$*, for some polynomial* poly*.*

Lemma 4.6 *We have* $\Pr[b' = 0 \mid b = 1] \leqslant \mathrm{negl}(\kappa)$*.*

4.2.1 Proof of Lemma 4.5.

To have a compact way to refer to Equation 8, for a given $(\mathsf{msk}, \mathsf{sk})$ let $\mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$ denote the distribution over secret-key vectors induced by $(\mathsf{msk}, \mathsf{sk})$, namely letting $\mathcal{D} := \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$,

$$\mathcal{D}(\mathbf{v}) := \Pr_{\mathbf{x} \leftarrow \mathbb{Z}_q^n}[\mathbf{x} = \mathbf{v} \mid \mathsf{sk} = \mathrm{KGen}(\mathsf{msk}, \mathbf{x})]. \tag{9}$$

Say msk is good if the probability of decryption correctness under msk is at least $\frac{47}{48}$. To have a compact way to denote this, let $\mathrm{PCorrect}(\mathsf{msk})$ denote the probability of correctness when all the parameters but msk are chosen at random.

$$\mathrm{PCorrect}(\mathsf{msk}) = \Pr_{\mathbf{u},\mathsf{sk},\mathbf{x}}[\mathrm{Dec}^O(\mathsf{sk}, \mathrm{Enc}^O(\mathsf{msk}, \mathbf{x})) = \langle \mathbf{u}, \mathbf{x} \rangle], \tag{10}$$

where $\mathbf{u}, \mathbf{x} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk} \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{u})$. We may extend this notation when additional parameters besides msk are fixed (e.g., $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$).

Equipped with this notation, we say msk is good if $\mathrm{PCorrect}(\mathsf{msk}) \geqslant \frac{47}{48}$. We make two claims and then show how to derive Lemma 4.5 from these two claims.

Claim 4.7 $\Pr_{\mathsf{msk} \leftarrow \mathrm{Setup}(1^\kappa)}[\mathsf{msk} \text{ is } \mathsf{good}\ ] \geqslant \frac{118}{6000}$.

Claim 4.8 Fix a good msk. For at least a $\frac{33}{60}$-fraction of $(\mathbf{u}, \mathsf{sk}[\mathbf{u}])$ produced under msk, the pair $(\mathsf{msk}, \mathsf{sk}[\mathbf{u}])$ is a binding to $\mathbf{u}$:

$$\Pr_{\mathbf{u},\mathsf{sk}[\mathbf{u}]}[\mathcal{D}(\mathbf{u}) > \frac{1}{2}] \geqslant \frac{33}{60}, \tag{11}$$

where $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk}[\mathbf{u}] \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{u})$ and $\mathcal{D} := \mathrm{Vect}(\mathsf{msk}, \mathsf{sk}[\mathbf{u}])$.

Proof of Lemma 4.5 via Claims 4.7 and 4.8. Let msk be the master secret key picked by the challenger to generate $\mathsf{sk}_i \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{w}_i)$ for $i \in [m]$. Since the challenge bit b is zero, each sk_i is indeed a secret key for $\mathbf{w}_i$ (as opposed to $\mathbf{u}_i$). Note that this msk is eventually picked by the adversary $\mathcal{A}$ because of the brute-force search in Step 1 of $\mathcal{A}$'s execution (i.e., $\mathcal{A}$ will eventually set $\widetilde{\mathsf{msk}} := \mathsf{msk}$ if it has not already output 0). Conditioned on msk being good, we show the probability that $\mathcal{A}$ outputs 0 is $1 - \mathrm{negl}(\kappa)$. Specifically, setting $\widetilde{\mathsf{msk}}$ to msk in Step 1's execution will make

$\mathcal{A}$ output 0 with all but negligible probability, assuming msk is good. This will prove the lemma because

$$\begin{aligned}\Pr[b'=0 \mid b=0] &\geqslant \Pr[b'=0 \wedge (\mathsf{msk} \text{ is } \mathsf{good}) \mid b=0] \\ &= \Pr[b'=0 \mid (b=0) \wedge (\mathsf{msk} \text{ is } \mathsf{good})] \Pr[\mathsf{msk} \text{ is } \mathsf{good} \mid b=0] \\ &= \Pr[b'=0 \mid (b=0) \wedge (\mathsf{msk} \text{ is } \mathsf{good})] \Pr[\mathsf{msk} \text{ is } \mathsf{good}] \\ &\geqslant (1-\mathrm{negl}(\kappa)) \times \frac{118}{6000}, \qquad \text{Claims 4.7}\end{aligned}$$

which is at least $1/\mathsf{poly}(\kappa)$ for some polynomial poly.

Let msk be the challenge master secret key picked by the challenger and suppose msk is good. Recall that for all i, $\mathsf{sk}_i \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{w}_i)$. Say $(\mathsf{msk}, \mathsf{sk})$ is a binding to $\mathbf{v}$ if $\mathcal{D}(\mathbf{v}) > \frac{1}{2}$, where $\mathcal{D} := \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$ (Equation 9). For $i \in [m]$, define a Bernoulli random variable X_i as follows: $X_i = 1$ iff $(\mathsf{msk}, \mathsf{sk}_i)$ is a binding to $\mathbf{w}_i$. Since each $\mathbf{w}_i$ is chosen uniformly at random from $\mathbb{Z}_q^n$ we have:

1. X_i's are i.i.d random variables.
2. $\mu := \Pr[X_1 = 1] \geqslant \frac{33}{60} \geqslant \frac{1}{q} + \frac{1}{\kappa}$ for all $q \geqslant 2$. This follows from Claim 4.8.

Let $X = \frac{X_1 + \cdots + X_m}{m}$. If $X \geqslant \frac{1}{q} + \frac{1}{2\kappa}$, then $\mathcal{A}$ will output 0. Let $\epsilon = \frac{1}{2\kappa}$. Since $m = 16(\kappa+\rho)\kappa^2$, we have $\frac{m\epsilon^2}{4} = \kappa + \rho$. Invoking the Chernoff bound (Lemma 3.1), the probability that $\mathcal{A}$ outputs 1 is at most

$$\Pr[X < \frac{1}{q} + \frac{1}{2\kappa}] \leqslant \Pr[|X - \mu| \geqslant \frac{1}{2\kappa}] \leqslant \frac{2}{e^{\frac{m\epsilon^2}{4}}} = \frac{2}{e^{\kappa+\rho}} = \mathrm{negl}(\kappa), \tag{12}$$

as desired. □

Proof of Claim 4.7. We show that if $\Pr_{\mathsf{msk} \leftarrow \mathrm{KGen}(1^\kappa)}[\mathsf{msk} \text{ is } \mathsf{good}\ ] = 118/6000 - \epsilon$ for $\epsilon > 0$, then the probability of correctness will be strictly less than 0.999, a contradiction. Let E be the event that the scheme decrypts correctly and let H be the event that the picked msk is good. We have $\Pr[E] < 47/48 + \Pr[H] \times 1$, which would be strictly less than 0.999, if $\Pr[H] < \frac{118}{6000}$. In particular, assuming $\Pr[H] < \frac{118}{6000}$, and since by the assumption of Lemma 4.4 we have $\frac{999}{1000} \leqslant \Pr[E]$:

$$\frac{999}{1000} \leqslant \Pr[E] < \frac{47}{48} + \frac{118}{6000} = \frac{23500}{24000} + \frac{472}{24000} = \frac{23972}{24000} < \frac{23976}{24000} = \frac{999}{1000}, \tag{13}$$

a contradiction. □

Proof of Claim 4.8. Fix a good msk for the rest of the proof.

Since msk is good, $\mathrm{PCorrect}(\mathsf{msk}) \geqslant \frac{47}{48} = \frac{11}{12} + \frac{1}{12} \cdot \frac{3}{4}$.

1. Say $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{correct}$ if $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}, \mathsf{sk}) > \frac{3}{4}$. Let $\chi := \Pr_{\mathbf{u}, \mathsf{sk}}[(\mathsf{msk}, \mathbf{u}, \mathsf{sk}) \text{ is } \mathsf{correct}]$. By an averaging argument, $\chi \geqslant \frac{11}{12}$. Because if $\chi = \frac{11}{12} - \epsilon$ for $\epsilon > 0$, then

$$\frac{11}{12} + \frac{1}{12}\frac{3}{4} = \frac{47}{48} \leqslant \mathrm{PCorrect}(\mathsf{msk}) \leqslant (\frac{11}{12} - \epsilon) \times 1 + (\frac{1}{12} + \epsilon)\frac{3}{4}, \tag{14}$$

implying that $\epsilon \leqslant \frac{3}{4}\epsilon$, a contradiction to $\epsilon > 0$.

2. If $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{correct}$, then for all $\mathbf{u}'$ where $\mathbf{u}' \in \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$ and $\mathbf{u}' \neq \mathbf{u}$, we have $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) < \frac{1}{4} + \frac{1}{q}$. In particular, since $\frac{1}{4} + \frac{1}{q} \leqslant \frac{3}{4}$ for all $q \geqslant 2$, $(\mathsf{msk}, \mathbf{u}', \mathsf{sk})$ cannot be $\mathsf{correct}$ for $\mathbf{u}' \neq \mathbf{u}$. To prove the statement, fix $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ and $\mathbf{u}'$ as above. Suppose toward a contradiction that for a random $\mathbf{x} \leftarrow \mathbb{Z}_q^n$, $\Pr_{\mathbf{x}}[\mathrm{Dec}^O(\mathsf{sk}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x})) = \langle \mathbf{u}', \mathbf{x} \rangle] \geqslant \frac{1}{4} + \frac{1}{q}$. We also know that $\Pr_{\mathbf{x}}[\mathrm{Dec}^O(\mathsf{sk}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x})) = \langle \mathbf{u}, \mathbf{x} \rangle] > \frac{3}{4}$. Thus, $\Pr_{\mathbf{x}}[\langle \mathbf{u}, \mathbf{x} \rangle = \langle \mathbf{u}', \mathbf{x} \rangle] > \frac{1}{q}$, a contradiction to $\mathbf{u} \neq \mathbf{u}'$.
3. Say $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{binding}$ if $\mathcal{D}(\mathbf{u}) > \frac{1}{2}$, where $\mathcal{D} = \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$; i.e., by reverse sampling a random vector based on $(\mathsf{msk}, \mathsf{sk})$, the vector $\mathbf{u}$ is picked with probability strictly greater than $\frac{1}{2}$.
4. We claim that at least a $\frac{3}{5}$ fraction of $\mathsf{correct}$ $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ are $\mathsf{binding}$. If proved, the probability that for a random $(\mathbf{u}, \mathsf{sk})$, $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{binding}$ is at least $\frac{11}{12} \times \frac{3}{5} = \frac{33}{60}$, as desired.

Suppose to the contrary that the statement in Item 4 does not hold, namely,

$$\Pr_{\mathbf{u} \leftarrow \mathbb{Z}_q^n, \mathsf{sk} \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{u})}[(\mathsf{msk}, \mathbf{u}, \mathsf{sk}) \text{ is } \mathsf{binding} \mid (\mathsf{msk}, \mathbf{u}, \mathsf{sk}) \text{ is } \mathsf{correct}] < \frac{3}{5}. \tag{15}$$

We claim that in this case $\chi = \Pr[(\mathsf{msk}, \mathbf{u}, \mathsf{sk}) \text{ is } \mathsf{correct}] < \frac{11}{12}$, where $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk} \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{u})$. This will be a contradiction because we earlier established that $\chi \geqslant \frac{11}{12}$.

To show $\chi < \frac{11}{12}$, we use a reverse-sampling technique. Given msk, we sample $(\mathbf{u}, \mathsf{sk})$ as usual, but we reverse-sample $\mathbf{u}'$ based on $(\mathsf{msk}, \mathsf{sk})$, and will instead evaluate the probability of $\Pr[(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) \text{ is } \mathsf{correct}]$. This latter probability should still equal χ, but we show that it is strictly smaller than $11/12$.

For the original $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$, where $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk} \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{u})$, let B and C be, respectively, the events that $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{binding}$ and $\mathsf{correct}$. Let $z = \Pr[B \mid C]$, and note that by Equation 15, $z < \frac{3}{5}$. Also, by Item 1 above, $\Pr[\overline{C}] \leqslant \frac{1}{12}$.

Sampling $\mathbf{u}' \leftarrow \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$ we have

$$\begin{aligned}
\chi &= \Pr[\overbrace{(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) \text{ is } \mathsf{correct}}^{D}] \\
&\leqslant \Pr[\overline{C}] \times \Pr[D \mid \overline{C}] + \Pr[B \wedge C] \times \Pr[D \mid B \wedge C] + \Pr[\overline{B} \wedge C] \Pr[D \mid \overline{B} \wedge C] \\
&\leqslant \Pr[\overline{C}] + \Pr[C] \Pr[B \mid C] \times \Pr[D \mid B \wedge C] + \Pr[C] \Pr[\overline{B} \mid C] \times \Pr[D \mid \overline{B} \wedge C] \\
&\leqslant \frac{1}{12} + \Pr[C] \Pr[B \mid C] + \chi(1 - z) Pr[D \mid \overline{B} \wedge C] \\
&\leqslant \frac{1}{12} + \chi z + (\chi \times (1 - z) \times \frac{1}{2}) \text{ (By Item 2, which implies } \Pr[D \mid \overline{B} \wedge C] \leqslant \frac{1}{2} \text{)} \\
&= \frac{1}{12} + \frac{\chi z}{2} + \frac{\chi}{2} \overset{(*)}{<} \frac{1}{12} + \frac{3}{5}\frac{\chi}{2} + \frac{\chi}{2} = \frac{1}{12} + \frac{8}{10}\chi.
\end{aligned}$$

The inequality marked with $(*)$ holds because $z < 3/5$.

This implies $\frac{1}{5}\chi < \frac{1}{12}$, implying $\chi < \frac{5}{12}$, which is a contradiction to the fact that $\chi \geqslant \frac{11}{12}$. The reason $\Pr[D \mid \overline{B} \wedge C] \leqslant \frac{1}{2}$ is that if C holds, then by Item 2, for all $\mathbf{u}' \neq \mathbf{u}$, the triple $(\mathsf{msk}, \mathbf{u}', \mathsf{sk})$ is not $\mathsf{correct}$. Also, since $\overline{B} \wedge C$ holds, the probability that $\mathbf{u}$ itself is picked (i.e., $\mathbf{u} = \mathbf{u}'$) is at most $1/2$. □

Proof of Lemma 4.6. Suppose $b = 1$ as in the lemma. We prove that for any fixed $\widetilde{\mathsf{msk}}$, the probability that that fixed $\widetilde{\mathsf{msk}}$ makes $\mathcal{A}$ output 0 is at most $\frac{2}{e^{\kappa+\rho}}$. The proof then follows by a

union bound over all $\mathsf{msk} \in \{0,1\}^\rho$. Fix $\widetilde{\mathsf{msk}}$. Let $(\mathbf{h}_1, \ldots, \mathbf{h}_m) = \mathrm{Recover}(\widetilde{\mathsf{msk}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$. Note that:

1. $(\mathbf{h}_1, \ldots, \mathbf{h}_m)$ is solely picked based on $(\mathbf{u}_1, \ldots, \mathbf{u}_m)$ (since $b = 1$), and not based on $(\mathbf{w}_1, \ldots, \mathbf{w}_m)$.
2. The Bernoulli random variables $(X_1, \ldots, X_m)$ are i.i.d, where $X_i = 1$ iff $\mathbf{w}_i = \mathbf{h}_i$.
3. For all $i \in [m]$, $\mu := \Pr[X_i = 1] \leqslant \frac{1}{q}$. This is because $|\mathsf{V}^\perp| = q$.

$\mathcal{A}$ outputs 0 iff $X := \frac{X_1 + \cdots + X_m}{m} \geqslant \frac{1}{q} + \frac{1}{2\kappa}$. Let $\epsilon = \frac{1}{2\kappa}$. Since $m = 16(\kappa + \rho)\kappa^2$, we have $\frac{m\epsilon^2}{4} = k + \rho$. Invoking the Chernoff bound (Lemma 3.1),

$$\Pr[X \geqslant \frac{1}{q} + \frac{1}{2\kappa}] \leqslant \Pr[|X - \mu| \geqslant \frac{1}{2\kappa}] \leqslant \frac{2}{e^{\frac{m\epsilon^2}{4}}} = \frac{2}{e^{\kappa+\rho}}, \tag{16}$$

as desired. □

Acknowledgments. Mohammad Hajiabadi was supported in part by the NSERC Discovery Grant RGPIN-2022-03270 and a Meta research award. Roman Langrehr was supported by SNSF Postdoc.Mobility grant no. 235425. Mingyuan Wang is supported in part by a research grant from Open Philanthropy.

References

[1] Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_19 2

[2] Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Berlin, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_33 2, 5, 9, 10, 17, 35

[3] Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_20 2, 4, 5

[4] 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 (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_16 2

[5] Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_21 2

[6] Acharya, O., Feng, W., Langrehr, R., O'Neill, A.: Access-controlled inner product function-revealing encryption. CiC **2**(3), 7 (2025). https://doi.org/10.62056/abe0zo-3y 2, 5

[7] Afshar, A., Couteau, G., Mahmoody, M., Sadeghi, E.: Fine-grained non-interactive key-exchange: Constructions and lower bounds. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part I. LNCS, vol. 14004, pp. 55–85. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30545-0_3 4, 12

[8] Agrawal, S., Clear, M., Frieder, O., Garg, S., O'Neill, A., Thaler, J.: Ad hoc multi-input functional encryption. In: Vidick, T. (ed.) ITCS 2020. vol. 151, pp. 40:1–40:41. LIPIcs (Jan 2020). https://doi.org/10.4230/LIPIcs.ITCS.2020.40 2

[9] Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_8 2, 4

[10] Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 34–64. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45374-9_2 2

[11] 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, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_12 2

[12] Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer, Berlin, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-47989-6_15 3

[13] 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 (Dec 2019). https://doi.org/10.1007/978-3-030-36030-6_8 5, 50

[14] Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_3 2, 58

[15] Barbosa, M., Catalano, D., Soleimanian, A., Warinschi, B.: Efficient function-hiding functional encryption: From inner-products to orthogonality. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 127–148. Springer, Cham (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_7 2

[16] Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 470–491. Springer, Berlin, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_20 2

[17] Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th FOCS. pp. 171–190. IEEE Computer Society Press (Oct 2015). https://doi.org/10.1109/FOCS.2015.20 3

[18] Boneh, D., Guan, J., Zhandry, M.: A lower bound on the length of signatures based on group actions and generic isogenies. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 507–531. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_18 4, 12

[19] Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_16 1

[20] Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46497-7_12 2, 5, 50

[21] Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 733–764. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_25 2

[22] Catalano, D., Fiore, D., Gennaro, R., Giunta, E.: On the impossibility of algebraic vector commitments in pairing-free groups. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part II. LNCS, vol. 13748, pp. 274–299. Springer, Cham (Nov 2022). https://doi.org/10.1007/978-3-031-22365-5_10 4, 12

[23] Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 164–195. Springer, Berlin, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-49384-7_7 2

[24] Döttling, N., Hartmann, D., Hofheinz, D., Kiltz, E., Schäge, S., Ursu, B.: On the impossibility of purely algebraic signatures. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 317–349. Springer, Cham (Nov 2021). https://doi.org/10.1007/978-3-030-90456-2_11 11, 12

[25] Garg, S., Srinivasan, A.: Single-key to multi-key functional encryption with polynomial loss. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 419–442. Springer, Berlin, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_16 3

[26] Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 95–120. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45374-9_4 2

[27] Hajiabadi, M., Langrehr, R., O'Neill, A., Wang, M.: On the black-box complexity of private-key inner-product functional encryption. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024, Part III. LNCS, vol. 15366, pp. 318–343. Springer, Cham (Dec 2024). https://doi.org/10.1007/978-3-031-78020-2_11 4, 12, 14, 57

[28] He, J., Li, S., Lin, W.K.: Lower bounds on inner-product functional encryption from all-or-nothing encryption primitives. In: Applebaum, B., Lin, H.R. (eds.) TCC 2025, Part II. LNCS, vol. 16269, pp. 123–152. Springer, Cham (Dec 2025). https://doi.org/10.1007/978-3-032-12293-3_5 12

[29] Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 544–562. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_29 2

[30] Li, B., Micciancio, D.: Compactness vs collusion resistance in functional encryption. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 443–468. Springer, Berlin, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_17 3

[31] Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: Dinur, I. (ed.) 57th FOCS. pp. 11–20. IEEE Computer Society Press (Oct 2016). https://doi.org/10.1109/FOCS.2016.11 54

[32] Luby, M., Rackoff, C.: How to construct pseudo-random permutations from pseudo-random functions (abstract). In: Williams, H.C. (ed.) CRYPTO'85. LNCS, vol. 218, p. 447. Springer, Berlin, Heidelberg (Aug 1986). https://doi.org/10.1007/3-540-39799-X_34 14, 40

[33] Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 1–12. Springer, Berlin, Heidelberg (Dec 2005). https://doi.org/10.1007/11586821_1 2

[34] O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010), https://eprint.iacr.org/2010/556 1

[35] Papakonstantinou, P.A., Rackoff, C.W., Vahlis, Y.: How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653 (2012), https://eprint.iacr.org/2012/653 6, 7, 8, 10, 12

[36] Rotem, L., Segev, G., Shahaf, I.: Generic-group delay functions require hidden-order groups. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 155–180. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45727-3_6 4, 11, 12

[37] Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Berlin, Heidelberg (May 2005). https://doi.org/10.1007/11426639_27 1

[38] Schul-Ganz, G., Segev, G.: Generic-group identity-based encryption: A tight impossibility result. In: Tessaro, S. (ed.) ITC 2021. LIPIcs, vol. 199, pp. 26:1–26:23. Schloss Dagstuhl (Jul 2021). https://doi.org/10.4230/LIPIcs.ITC.2021.26 4, 6, 7, 8, 10, 12, 20

[39] Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_27 2

[40] Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Berlin, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_18 2, 3, 12, 33

[41] Tairi, E., Ünal, A.: Lower bounds for lattice-based compact functional encryption. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 249–279. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_9 2

[42] Tomida, J.: Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 459–488. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_16 2

[43] Tomida, J.: Unbounded quadratic functional encryption and more from pairings. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 543–572. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_18 2

[44] Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 408–425. Springer, Cham (Sep 2016). https://doi.org/10.1007/978-3-319-45871-7_24 2

[45] Ünal, A.: Impossibility results for lattice-based functional encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 169–199. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_7 2

[46] Wee, H.: Functional encryption for quadratic functions from k-lin, revisited. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 210–228. Springer, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64375-1_8 2

[47] Zhandry, M.: To label, or not to label (in generic groups). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 66–96. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_3 3, 11, 12, 33

A Maurer's Generic Group Model (MGGM)

Definition A.1 (Maurer's GGM). An algorithm $\mathsf{Alg}^{\mathbb{G}}(1^\kappa, T)$ in Maurer's GGM (MGGM) $\mathbb{G}$ receives as input a table $T\colon \mathbb{N}_+ \to \mathbb{Z}_p \cup \{\bot\}$.[8] For each algorithm there has to exist a large enough polynomial $f(\kappa)$, such that we have $T(i) = \bot$ for all $i \geqslant f(\kappa)$.[9] $T(1)$ will always store 1. The algorithm Alg may *only* read or manipulate the table via the following oracles $\mathbb{G} := (\mathsf{Add}, \mathsf{Eql})$.

- **Addition queries** Add: The algorithm Alg makes a query of the form $(a_1, a_2, i_1, i_2, i_3) \in \mathbb{Z}_p^2 \times \mathbb{N}_+^3$. If $T(i_1) = \bot$, $T(i_2) = \bot$, or $i_3 = 1$, then nothing happens. Else, assuming $T(i_1) = x_1$ and $T(i_2) = x_2$, then $T(i_3)$ is set to $a_1 x_1 + a_2 x_2$.
- **Equality queries** Eql: The algorithm Alg makes a query of the form $(i_1, i_2) \in \mathbb{Z}^2$. In return, Alg receives 1 if $T(i_1) \neq \bot$ and $T(i_1) = T(i_2)$, and receives 0, otherwise.

Output: Finally, Alg outputs a non-group element $a \in \{0,1\}^*$ (which could be ϵ) together with a sequence of t group elements, for an arbitrary $t \geqslant 1$. We capture the latter by assuming that (the DLOG of) the ith output group element is written at $T(i)$, and that $T(j) = \bot$, for all $j \geqslant t+1$. If $t = 1$, we say Alg outputs no group elements;[10] else, we say Alg's output involves group elements.
Input: If Alg has no group elements in its input, it receives a table T with $T(1) = 1$ and $T(i) = \bot$ for all $i \geqslant 2$. If its input comprises of the outputs of two or more MGGM algorithms (e.g., a decryption algorithm receiving both secret keys and ciphertexts), their tables are merged into a

[8] Note that the group order p does not necessarily need to agree with the q for the inner product functionality.
[9] Since the table is defined over an infinite domain $\mathbb{N}_+$, this simplification is essentially saying that all algorithms will only work with the first $f(\kappa)$ entries of the table.
[10] Technically, it still outputs the generator that is stored in $T(1)$, but we ignore this minor technicality.

single table, and the algorithm also receives input indicating which table entries correspond to which part of the input.

Intuitively, this table T contains the group elements that the algorithm has access to, where i is the "handle" to the group element and $T(i)$ stores the discrete log of this element. We emphasize that, even though the algorithm takes T as input, it cannot simply read off the value of $T(i)$. We will use in the following $g_1, \ldots, g_m$ to denote handles to group elements and use additive notation for group operations, that are implicitly realized via the Add oracle.

We write $\mathsf{Alg}^{\mathbb{G}}$ to denote that Alg has access to both Add and Eql, and gets the group order p implicitly as additional input. A scheme in MGGM is a list of algorithms in MGGM. We make the following remark.

Remark A.2 For algorithms $\mathsf{Alg}^{\mathbb{G}}(g_1, \ldots, g_m)$ that never output group elements, we may assume without loss of generality that they never make Add queries and instead use only an Eql^* oracle that accepts inputs of the form $(i_1, \ldots, i_m, z_1, \ldots, z_m) \in \mathbb{Z}^m \times \mathbb{Z}_p^m$ for an arbitrary $m \in \mathbb{N}_+$ and returns 1 if $\sum_{j=1}^m z_j \cdot T(i_j) = 0$ and for all $j \in [m]$ $T(i_j) \neq \perp$. Otherwise, it returns 0.

Throughout this paper, we use (additive) group notation to denote the operations performed using the Add and Eql oracle, instead of working directly with the tables and indices. We use g to denote the generator of the group.

A.1 Attacking the Variant of the Decryption Algorithm

We highlight the changes we need to make to Lemma 4.4 to adapt it to the two-phase decryption algorithm of Definition 3.5. First of all, we need to change the definition of $\mathrm{PCorrect}(\mathsf{msk})$ (Equation 10) accordingly for this setting. $\mathrm{PCorrect}(\mathsf{msk})$ is the largest value α such that

$$\Pr_{\mathbf{u},\mathsf{sk},\mathbf{x}}\Big[\mathrm{Dec}^O(\mathsf{sk}, \mathrm{Enc}^O(\mathsf{msk}, \mathbf{x}), \langle \mathbf{u}, \mathbf{x}\rangle) = 1)\Big] \geqslant \alpha$$
$$\Pr_{\mathbf{u},\mathsf{sk},\mathbf{x}}\Big[\mathrm{Dec}^O(\mathsf{sk}, \mathrm{Enc}^O(\mathsf{msk}, \mathbf{x}), \langle \mathbf{u}, \mathbf{x}\rangle + z) = 0)\Big] \geqslant \alpha, \qquad \text{for all } z \in \mathbb{Z}_q \setminus \{0\},$$

where $\mathbf{u}, \mathbf{x} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk} \leftarrow \mathrm{KGen}(\mathsf{msk}, \mathbf{u})$. We may define notation such as $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$, when all of $\mathsf{msk}, \mathbf{u}, \mathsf{sk}$ are fixed, in the natural way.

Now all parts of the proofs will remain the same. The only thing we have to update is the fact established in Item 2, namely that If $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{correct}$, then for all $\mathbf{u}'$ where $\mathbf{u}' \in \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$ and $\mathbf{u}' \neq \mathbf{u}$, $(\mathsf{msk}, \mathbf{u}', \mathsf{sk})$ cannot be $\mathsf{correct}$. We argue this as follows.

We claim if $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{correct}$, then for all $\mathbf{u}'$ where $\mathbf{u}' \in \mathrm{Vect}(\mathsf{msk}, \mathsf{sk})$ and $\mathbf{u}' \neq \mathbf{u}$, we have $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) \leqslant \frac{3}{4}$. This implies that $(\mathsf{msk}, \mathbf{u}', \mathsf{sk})$ cannot be $\mathsf{correct}$ for $\mathbf{u}' \neq \mathbf{u}$. To prove the statement, fix $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ and $\mathbf{u}'$ as above, and suppose to the contrary that $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) > \frac{3}{4}$. Since $(\mathsf{msk}, \mathbf{u}, \mathsf{sk})$ is $\mathsf{correct}$,

$$\Pr_{\mathbf{x}}[\mathrm{Dec}(\mathsf{sk}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}), \langle \mathbf{u}, \mathbf{x}\rangle) = \top] > \frac{3}{4}. \tag{17}$$

Letting $\mathbf{d} = \mathbf{u} - \mathbf{u}'$

$$\Pr_{\mathbf{x}}[\mathrm{Dec}(\mathsf{sk}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}), \langle \mathbf{u}', \mathbf{x}\rangle + \langle \mathbf{d}, \mathbf{x}\rangle) = \top] > \frac{3}{4}. \tag{18}$$

Since $\Pr_{\mathbf{x}}[\langle \mathbf{d}, \mathbf{x}\rangle = 0] = \frac{1}{q} \leqslant \frac{1}{2}$,

$$\Pr_{\mathbf{x}}\big[(\langle \mathbf{d}, \mathbf{x}\rangle \neq 0) \wedge \mathrm{Dec}\big(\mathsf{sk}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}), \langle \mathbf{u}', \mathbf{x}\rangle + \langle \mathbf{d}, \mathbf{x}\rangle\big) = \top\big] > \frac{1}{4}, \tag{19}$$

which is a contradiction to $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) > \frac{3}{4}$. Because $\mathrm{PCorrect}(\mathsf{msk}, \mathbf{u}', \mathsf{sk}) > \frac{3}{4}$ implies that for all $z \neq 0$,

$$\Pr_{\mathbf{x}}\big[\mathrm{Dec}\big(\mathsf{sk}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}), \langle \mathbf{u}', \mathbf{x}\rangle + z\big) = \top\big] < \frac{1}{4}, \tag{20}$$

a contradiction to Equation 19.

B Shoup's Generic Group Model

We begin by recalling Shoup's GGM oracles.

Definition B.1 (Shoup's GGM [40, 47]). An SGGM oracle $\overline{\mathbb{G}}$ consists of two oracles $\overline{\mathbb{G}} := (\mathsf{Label}, \mathsf{SAdd})$, parametrized by a security parameter and an associated prime number $p = p(\kappa) \in \{0,1\}^\kappa$ with $p \geqslant 2^{\kappa-1}$. We describe the oracles for a fixed security parameter κ. We omit the mention of the security parameter κ as an input to the oracles for better readability. The oracles are defined with respect to an injective label function $\mathsf{L}\colon \mathbb{Z}_p \mapsto \mathsf{LB}$ and label space $\mathsf{LB} := \mathsf{LB}_\kappa \subseteq \{0,1\}^{t(\kappa)}$ for some fixed polynomial t.

- **Label queries** Label**:** A $\mathsf{Label}(x \in \mathbb{Z}_p)$ query is answered with $\mathsf{L}(x)$.
- **Addition queries** SAdd**:** A $\mathsf{SAdd}(a_0, a_1, \mathsf{lb}_0, \mathsf{lb}_1)$ query outputs lb_3, computed as follows. If $\exists x_0, x_1 \in \{0,1\}^\kappa$ such that $\mathsf{L}(x_0) = \mathsf{lb}_0$ and $\mathsf{L}(x_1) = \mathsf{lb}_1$, set $\mathsf{lb}_3 := \mathsf{L}((a_0 x_0 + a_1 x_1) \mod p)$. Else, set $\mathsf{lb}_3 := \perp$.

An algorithm in Shoup's GGM is an algorithm with oracle access to $\overline{\mathbb{G}}$ which is also given the order of the group p as additional input. We do not write the input p explicitly throughout the paper.

A scheme in Shoup's GGM is a list of algorithms that all get access to $\overline{\mathbb{G}}$ for the same function L, which is initially chosen uniformly at random from the set of all injective functions from $\mathbb{Z}_p$ to LB.

For our impossibility in Shoup's model, we do not use the above oracle model directly. Instead, we use the following variant which captures the algebraic structure of the group as in Maurer's GGM and the random representations for group elements with a separate oracle. This model allows us to extend our impossibility result in Maurer's GGM to Shoup's GGM in a modular way. Our new definition is novel, and may help extend the previous Maurer impossibility results to Shoup's in a seamless fashion, as carried out in our paper.

Definition B.2 An algorithm $\mathsf{Alg}^{\mathbb{G},\mathbb{L}}(1^\kappa, T)$ in Maurer's GGM with Random Labels/Representations in $\mathsf{LB} \subseteq \{0,1\}^*$ — referred to as *RL-MGGM* — is defined similarly to an algorithm in Maurer's GGM (Definition A.1) with oracle access to $\mathbb{G}$. In addition, it is given access to the oracles $\mathbb{L} := (\mathsf{PointerToLabel}, \mathsf{LabelToPointer})$. As in SGGM, these oracles are defined relative to a label function $\mathsf{L}\colon \mathbb{Z}_p \mapsto \mathsf{LB}$ sampled uniformly at random from the set of all injective functions from $\mathbb{Z}_p$ to LB.

- **Label queries** $\mathsf{PointerToLabel}$**:** An $\mathsf{PointerToLabel}(i \in \mathbb{Z})$ query is answered with $\mathsf{L}(T(i))$, if $T(i) \neq \perp$, and with $\perp$ otherwise.

- **Index queries** LabelToPointer**:** A LabelToPointer$(\mathsf{lb}, i \in \mathbb{Z})$ query does nothing if lb is not in the image of L or i is not a valid index for the table T. Otherwise, this query stores $\mathsf{L}^{-1}(\mathsf{lb})$ in $T(i)$.

A scheme in RL-MGGM is a list of algorithms in RL-MGGM that all get access to PointerToLabel and LabelToPointer for the same function L, which is initially chosen uniformly at random from the set of all injective functions from $\mathbb{Z}_p$ to LB.

Theorem B.3 (Simulating SGGM via RL-MGGM). *A scheme Π in Shoup's GGM can be simulated by a scheme Π' in RL-MGGM. This simulation preserves the input/output behavior, the runtime up to a constant overhead for each query and increases the number of queries at most by a factor of 4. Moreover, if an adversary $\mathcal{A}$ breaks Π' in the RL-MGGM then there exists an adversary $\mathcal{B}$ that breaks Π in Shoup's GGM with the same advantage, essentially the same runtime, and the same number of queries.*

Proof. We show how to convert an algorithm in Shoup's GGM with respect to a label function L to an algorithm in Maurer's GGM with random representation with respect to the same label function L. Since schemes in Shoup's GGM and in Maurer's GGM with random representations choose the function L according to the same distribution, this is sufficient to prove the theorem.

Let Alg be an algorithm in Shoup's GGM. Then we construct an algorithm Alg' in Maurer's GGM with random labels that perfectly simulates Alg and, in particular, only outputs a bitstring and no MGGM table. The algorithm Alg' runs Alg on its input and returns its output. We simulate the oracle queries of Alg as follows.

- **Label queries** Label**:** A Label$(x \in \mathbb{Z}_p)$ query is answered by first computing g^x using the Add oracle and then returning PointerToLabel(g^x).
- **Shoup addition queries** SAdd**:** A SAdd$(a_0, a_1, \mathsf{lb}_0, \mathsf{lb}_1)$ query outputs lb_3, computed as follows. First, we make a LabelToPointer(lb_0, g_1) and a LabelToPointer(lb_1, g_2) query to obtain (pointers to the) group elements g_1, g_2. Secondly, we make an Add query to obtain $a_0 \cdot g_1 + a_1 \cdot g_2$. Finally, we make an PointerToLabel$(g_1 + g_2)$ query and return the resulting label.

It is clear that these simulated queries behave identically to Shoup's GGM.

Any cryptographic construction in SGGM can be translated into a cryptographic construction in the RL-MGGM model, where the algorithms involved do not output any MGGM tables, by applying the simulation strategy from above to each algorithm of the scheme. This allows adversaries against the resulting RL-MGGM-based construction to be converted to adversaries against the base SGGM-based construction. The reduction is possible because none of the algorithms in the RL-MGGM-based construction output MGGM tables, and thus the input to an RL-MGGM adversary does not contain any tables either.[11] □

A consequence of this theorem is that impossibility in RL-GGM implies impossibility in SGGM.

To improve readability, we write PointerToLabel(g_1) = Label for a PointerToLabel query whose input is a pointer to the group element g_1 and whose output is Label. Similary, we write LabelToPointer(Label) $= g_1$ for a LabelToPointer(Label, i) query that stores the group element g_1 in $T(i)$.

[11] Note that if the input to the RL-MGGM adversary contains an MGGM table, it becomes unclear how to simulate such an adversary using a Shoup adversary, since a Shoup adversary cannot answer table-related equality queries.

C Impossibility of FH-IPFE in Shoup's GGM

In this section, we extend our impossibility result from MGGM to Shoup's model (SGGM). By virtue of Theorem B.3, it suffices to prove the impossibility in the RR-GGM model. Recall that in this model the algorithms are provided with an oracle $(\mathbb{G}, \mathbb{L})$, described in Definition B.2.

C.1 New Security Notions

Definition C.1 We call $\mathsf{U}, \mathsf{V} \subseteq \mathbb{Z}_q^n$ with $\mathsf{dim}(\mathsf{V}) = n-1$, $\mathsf{dim}(\mathsf{U}) = n-2$, and $\mathsf{V}^\perp \nsubseteq \mathsf{U}$ *a valid pair*.

Next, we introduce two new security notions that are parameterized by a valid pair U, V. In Appendix D we show that the standard FH-IND-CPA security notion implies both of these notions for any V. Thus, our result also rules out FH-IND-CPA secure IPFEs.

Definition C.2 ((U, V)-key hiding). Let $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$ be an IPFE over $\mathbb{Z}_q^n$. The game is defined over a PPT adversary $\mathcal{A}$ and a challenger. The challenger samples a master secret key $\mathsf{msk} \leftarrow \mathrm{Setup}^O(1^\kappa)$ and a challenge bit $b \leftarrow \{0,1\}$. The adversary $\mathcal{A}$, who has the description of U and V, can make queries of the following types, where neither type requires an input from $\mathcal{A}$, and where all the queries must be made non-adaptively at once.

- **Straight key corruption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{u}, \mathrm{KGen}(\mathsf{msk}, \mathbf{u}))$, where $\mathbf{u} \leftarrow \mathsf{U}$.
- **LoR key corruption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{y}, \mathbf{y}+\mathbf{w}, \mathrm{KGen}(\mathsf{msk}, \mathbf{y}+b\mathbf{w}))$, where $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{w} \leftarrow \mathsf{V}^\perp$.
- **Encryption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{x}, \mathrm{Enc}(\mathsf{msk}, \mathbf{x}))$, where $\mathbf{x} \leftarrow \mathsf{V}$.

At the end, $\mathcal{A}$ outputs a bit b' and wins if $b = b'$. We say that $\mathcal{A}$ is admissible if all the adversary's queries are made non-adaptively at once. We say that $\mathcal{E}$ is V-key hiding if any admissible PPT $\mathcal{A}$ wins the game with probability at most $1/2 + \mathrm{negl}(\kappa)$.

Note that the above definition can be realized under the DDH assumption, since it only concerns key hiding, and not plaintext hiding. In particular, the dual version of any IPFE that satisfies the standard notion of IND-CPA security satisfies the above notion (e.g., [2]). Dual means swapping the roles of the key-generation and encryption algorithms. Thus, the above definition, in isolation, cannot be ruled out in the GGM. Thus, we give a definition for plaintext hiding below, and will show that these two definitions cannot be simultaneously realized in the GGM.

Definition C.3 ((U, V)-plaintext hiding). Let $\mathsf{V} \subseteq \mathbb{Z}_q^n$ be an $(n-1)$-dimensional subspace of $\mathbb{Z}_q^n$ and $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$ be an IPFE over $\mathbb{Z}_q^n$. Let msk and bit b be sampled as in Definition 3.3. The adversary $\mathcal{A}$ may make queries of the following types, all made non-adaptively at once.

- **Key corruption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{u}, \mathrm{KGen}(\mathsf{msk}, \mathbf{u}))$, where $\mathbf{u} \leftarrow \mathsf{U}$.
- **Special key corruption query**: $\mathcal{A}$ receives $(\mathbf{y}, \mathrm{KGen}(\mathsf{msk}, \mathbf{y}))$, where $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ conditioned on $(\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^\perp \nsubseteq \mathsf{V}$. The adversary may make this query at most once.
- **Straight encryption queries**: For any such query, $\mathcal{A}$ receives $(\mathbf{v}, \mathrm{Enc}(\mathsf{msk}, \mathbf{v}))$, where $\mathbf{v} \leftarrow \mathsf{V}$.

- **LoR encryption queries**: For any such query, $\mathcal{A}$ receives $(\langle \mathbf{y}, \mathbf{v} + b\mathbf{x}\rangle, \mathrm{Enc}(\mathsf{msk}, \mathbf{v} + b\mathbf{x}))$, where $\mathbf{v} \leftarrow \mathsf{V}$ and $\mathbf{x} \leftarrow \mathbb{Z}_q^n$.

At the end, $\mathcal{A}$ outputs a bit b' and wins if $b = b'$. We say that $\mathcal{A}$ is admissible if $\mathcal{A}$ makes at most one key corruption query and if all of $\mathcal{A}$'s queries are made non-adaptively at once. We say that $\mathcal{E}$ is V-plaintext hiding if any admissible PPT $\mathcal{A}$ wins the game with probability at most $1/2 + \mathrm{negl}(\kappa)$.

C.2 Step 1: Removing Label-Queries from Dec in RL-MGGM

In Figure 3 we present a compiler that removes PointerToLabel and LabelToPointer queries from Dec in an IPFE in RL-MGGM.

Lemma C.4 *Assume* $\mathcal{E}^{\mathbb{G},\mathbb{L}} := (\mathrm{Setup}, \mathrm{KGen}^{\mathbb{G},\mathbb{L}}, \mathrm{Enc}^{\mathbb{G},\mathbb{L}}, \mathrm{Dec}^{\mathbb{G},\mathbb{L}})$ *is an IPFE for* $\mathbb{Z}_q^n$ *where*

- $n \geqslant 4$ *and*
- q^n *is superpolynomial.*

Assume that $\mathcal{E}^{\mathbb{G},\mathbb{L}}$ *is* ν*-correct and satisfies both* (U,V)*-key hiding (Definition C.2) and* (U,V)*-plaintext hiding (Definition C.3) for subspaces* $\mathsf{U},\mathsf{V} \subseteq \mathbb{Z}_q^n$ *sampled uniformly at random conditioned on satisfying Definition C.1.*

Then the compiled schemes $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}} := (\overline{\mathrm{Setup}}^{\mathbb{G},\mathbb{L}}, \overline{\mathrm{KGen}}^{\mathbb{G},\mathbb{L}}, \overline{\mathrm{Enc}}^{\mathbb{G},\mathbb{L}}, \overline{\mathrm{Dec}}^{\mathbb{G}})$ *defined in Figure 3 continue to be PPT algorithms, and*

1. *for every polynomial* poly *we can set the parameters of the compiler so that* $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$ *is* $\nu - 1/\mathsf{poly}(\kappa)$*-correct,*
2. *for the subspaces* $\mathsf{U},\mathsf{V} \subseteq \mathbb{Z}_q^n$ *sampled during* $\overline{\mathrm{Setup}}^{\mathbb{G},\mathbb{L}}(1^\kappa)$ *the scheme is*
   (a) $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$ *is* (U,V)*-key hiding and*
   (b) $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$ *is* (U,V)*-plaintext hiding.*

Proof of Lemma C.4.

Correctness. Let $\overline{\mathsf{msk}} = (\mathsf{msk}, (\mathbf{v}_1, \mathsf{ct}_1, \ldots, \mathbf{v}_{\ell_\mathsf{e}}, \mathsf{ct}_{\ell_\mathsf{e}}), (\mathbf{u}_1, \mathsf{sk}_1, \ldots, \mathbf{u}_{\ell_\mathsf{k}}, \mathsf{sk}_{\ell_\mathsf{k}})) \leftarrow \overline{\mathrm{Setup}}^{\mathbb{G},\mathbb{L}}(1^\kappa)$, $\mathbf{x} \leftarrow \mathbb{Z}_q^n$, $\mathbf{y} \leftarrow \mathbb{Z}_q^n$, $\overline{\mathsf{sk}}[\mathbf{y}] = (\mathsf{sk}[\mathbf{y}], L_\mathsf{k}) \leftarrow \overline{\mathrm{KGen}}^{\mathbb{G},\mathbb{L}}(\overline{\mathsf{msk}}, \mathbf{y})$, and $\overline{\mathsf{ct}}[\mathbf{x}] = (\mathsf{ct}[\mathbf{x}], L_\mathsf{e}) \leftarrow \overline{\mathrm{Enc}}^{\mathbb{G},\mathbb{L}}(\overline{\mathsf{msk}}, \mathbf{x})$.

Define the following sets of exponent/label pairs used during encryption, key generation, and decryption.

$$Q_\mathsf{e} := \{(z, \mathsf{Label}) \mid \mathrm{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x}) \text{ makes a } \mathsf{PointerToLabel}(z \cdot g) = \mathsf{Label} \text{ or a } \mathsf{LabelToPointer}(\mathsf{Label}) = z \cdot g \text{ query.}\}$$

$$Q_\mathsf{k} := \{(z, \mathsf{Label}) \mid \mathrm{KGen}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{y}) \text{ makes a } \mathsf{PointerToLabel}(z \cdot g) = \mathsf{Label} \text{ or a } \mathsf{LabelToPointer}(\mathsf{Label}) = z \cdot g \text{ query.}\}$$

$$Q_\mathsf{d} := \{(z, \mathsf{Label}) \mid \mathrm{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}]) \text{ makes a } \mathsf{PointerToLabel}(z \cdot g) = \mathsf{Label} \text{ or a } \mathsf{LabelToPointer}(\mathsf{Label}) = z \cdot g \text{ query.}\}$$

Moreover, set $\overline{Q}_\mathsf{e} := \{(z, \mathsf{Label}) \mid (z \cdot g, \mathsf{Label}) \in L_\mathsf{e}\}$ and $\overline{Q}_\mathsf{k} := \{(z, \mathsf{Label}) \mid (z \cdot g, \mathsf{Label}) \in L_\mathsf{k}\}$.

We define the following bad events. The events $\mathsf{Intersection}_\mathsf{e}$ and $\mathsf{Intersection}_\mathsf{k}$ capture the scenario where $\overline{\mathrm{Dec}}^{\mathbb{G}}$ simulates a $\mathbb{L}$ query inconsistently with a query made during $\mathrm{KGen}^{\mathbb{G},\mathbb{L}}$ or $\mathrm{Enc}^{\mathbb{G},\mathbb{L}}$. The events $\mathsf{CollG}_\mathsf{e}$, $\mathsf{CollG}_\mathsf{k}$, $\mathsf{CollL}_\mathsf{e}$, and $\mathsf{CollL}_\mathsf{k}$ where the $\mathbb{L}$ queries simulated during $\overline{\mathrm{Dec}}^{\mathbb{G}}$ combined with the $\mathbb{L}$ queries made during $\mathrm{KGen}^{\mathbb{G},\mathbb{L}}$ or $\mathrm{Enc}^{\mathbb{G},\mathbb{L}}$ assign the same label to two different group elements or two different labels to the same group element.

$\mathsf{Intersection}_\mathsf{e}$**:** $(Q_\mathsf{d} \setminus L_\mathsf{e}) \cap Q_\mathsf{e} \neq \emptyset$.

$\overline{\text{Setup}}^{\mathbb{G},\mathbb{L}}(1^\kappa)$:

1. Sample msk according to $\text{Setup}\,1^\kappa)$.
2. Sample a valid pair (U,V) (see Definition C.1) uniformly at random.

Let $\ell_\mathsf{e}, \ell_\mathsf{k} \in \mathbb{N}_+$ be integers which will be instantiated later (refer to the proof for the correctness part).

3. For $i \in [\ell_\mathsf{e}]$, sample $\mathbf{v}_i \leftarrow \mathsf{V}$ and $\mathsf{ct}_i \leftarrow \text{Enc}^{\mathbb{G}}(\mathsf{msk}, \mathbf{v}_i)$.
4. For $i \in [\ell_\mathsf{k}]$, sample $\mathbf{u}_i \leftarrow \mathsf{U}$ and $\mathsf{sk}_i \leftarrow \text{KGen}^{\mathbb{G}}(\mathsf{msk}, \mathbf{u}_i)$.

Let the resulting master secret key $\overline{\mathsf{msk}}$ contain msk and $\mathsf{pp} := ((\mathbf{v}_1, \mathsf{ct}_1, \ldots, \mathbf{v}_{\ell_\mathsf{e}}, \mathsf{ct}_{\ell_\mathsf{e}}), (\mathbf{u}_1, \mathsf{sk}_1, \ldots, \mathbf{u}_{\ell_\mathsf{k}}, \mathsf{sk}_{\ell_\mathsf{k}}))$.

$\overline{\text{KGen}}^{\mathbb{G},\mathbb{L}}(\overline{\mathsf{msk}} := (\mathsf{msk}, \mathsf{pp}), \mathbf{y})$**:**

1. Sample $\mathsf{sk}[\mathbf{y}] \leftarrow \text{KGen}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{y})$.
2. Initialize a table $L_\mathsf{k} := \emptyset$.
3. For all $i \in [\ell_\mathsf{e}]$, perform $\text{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}_i, \langle \mathbf{y}, \mathbf{v}_i \rangle)$ (where $\mathbf{v}_i$ and ct_i are obtained from pp). For all $\mathsf{PointerToLabel}(g_1)$ queries answered with Label and all $\mathsf{LabelToPointer}(\mathsf{Label})$ queries answered with g_1 during decryption, add (g_1, Label) to the table L_k.[a]

Let the resulting secret key $\overline{\mathsf{sk}}[\mathbf{y}]$ contain $\mathsf{sk}[\mathbf{y}]$ and the table L_k.

$\overline{\text{Enc}}^{\mathbb{G},\mathbb{L}}(\overline{\mathsf{msk}} := (\mathsf{msk}, \mathsf{pp}), \mathbf{x})$**:**

1. Sample $\mathsf{ct}[\mathbf{x}] \leftarrow \text{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x})$.
2. Initialize a table $L_\mathsf{e} := \emptyset$.
3. For all $i \in [\ell_\mathsf{k}]$, perform $\text{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{sk}_i, \mathsf{ct}[\mathbf{x}], \langle \mathbf{u}_i, \mathbf{x} \rangle)$ (where $\mathbf{u}_i$ and sk_i are obtained from pp). For all $\mathsf{PointerToLabel}(g_1)$ queries answered with Label and all $\mathsf{LabelToPointer}(\mathsf{Label})$ queries answered with g_1 during decryption add (g_1, Label) to the table L_e.

Let the resulting ciphertext $\overline{\mathsf{ct}}[\mathbf{x}]$ contain $\mathsf{ct}[\mathbf{x}]$ and the table L_e.

$\overline{\text{Dec}}^{\mathbb{G}}\left(\overline{\mathsf{sk}[\mathbf{y}]} = (\mathsf{sk}[\mathbf{y}], L_\mathsf{k}), \overline{\mathsf{ct}[\mathbf{x}]} = (\mathsf{ct}[\mathbf{x}], L_\mathsf{e}), z\right)$**:**

1. Initialize a table $L_\mathsf{d} := L_\mathsf{e} \cup L_\mathsf{k}$.
2. Return $\text{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], z)$ while simulating $\mathbb{L}$ as follows.
   (a) $\mathsf{PointerToLabel}(g_1)$ **queries:** Search L_d for an entry of the form (g_1, Label). If such an entry is found, return Label. Otherwise, sample a new label $\mathsf{Label} \leftarrow \mathsf{LB}$, add (g_1, Label) to L_d and return Label.
   (b) $\mathsf{LabelToPointer}(\mathsf{Label})$ **queries:** If $\mathsf{Label} \notin \mathsf{LB}$ do nothing. Otherwise, search L_d for an entry of the form (g_1, Label). If such an entry is found return g_1. Otherwise, with probability $1 - p/|\mathsf{LB}|$, store $(\perp, \mathsf{Label})$ in L_d. In the remaining case, sample $x \leftarrow \mathbb{Z}_q$ and store $(x \cdot g, \mathsf{Label})$ to L_d.

[a] Formally, we store a copy of the group element and a table index i' that is not touched by any other algorithm and add (i', Label) to L_k. The group elements stored in the indices stored in L_k are always output along with L_k.

Figure 3: The compiler for Lemma C.4 that compiles an IPFE $(\text{Setup}, \text{KGen}^{\mathbb{G},\mathbb{L}}, \text{Enc}^{\mathbb{G},\mathbb{L}}, \text{Dec}^{\mathbb{G},\mathbb{L}})$ into an IPFE $(\overline{\text{Setup}}^{\mathbb{G},\mathbb{L}}, \overline{\text{KGen}}^{\mathbb{G},\mathbb{L}}, \overline{\text{Enc}}^{\mathbb{G},\mathbb{L}}, \overline{\text{Dec}}^{\mathbb{G}})$. Recall that $\mathbb{G}$ is Maurer's GGM oracle and $\mathbb{L}$ is the oracle handling the $\mathsf{PointerToLabel}$ and $\mathsf{LabelToPointer}$ queries.

$\mathsf{Intersection_k}$: $(Q_\mathsf{d} \setminus L_\mathsf{k}) \cap Q_\mathsf{k} \neq \emptyset$.

$\mathsf{CollG_e}$: $\exists z_1, z_2 \in \mathbb{Z}_p, \mathsf{Label} \in \mathsf{LB} : (z_1, \mathsf{Label}) \in Q_\mathsf{e} \wedge (z_2, \mathsf{Label}) \in Q_\mathsf{d}$

$\mathsf{CollG_k}$: $\exists z_1, z_2 \in \mathbb{Z}_p, \mathsf{Label} \in \mathsf{LB} : (z_1, \mathsf{Label}) \in Q_\mathsf{k} \wedge (z_2, \mathsf{Label}) \in Q_\mathsf{d}$

$\mathsf{CollL_e}$: $\exists z \in \mathbb{Z}_p, \mathsf{Label}_1, \mathsf{Label}_2 \in \mathsf{LB} : (z, \mathsf{Label}_1) \in Q_\mathsf{e} \wedge (z, \mathsf{Label}_2) \in Q_\mathsf{d}$

$\mathsf{CollL_k}$: $\exists z \in \mathbb{Z}_p, \mathsf{Label}_1, \mathsf{Label}_2 \in \mathsf{LB} : (z, \mathsf{Label}_1) \in Q_\mathsf{k} \wedge (z, \mathsf{Label}_2) \in Q_\mathsf{d}$

$\mathsf{Bad} := \mathsf{Intersection_e} \vee \mathsf{Intersection_k} \vee \mathsf{CollG_e} \vee \mathsf{CollG_k} \vee \mathsf{CollL_e} \vee \mathsf{CollL_k}$

Claim C.5 $\Pr\left[\overline{\mathrm{Dec}}^{\mathbb{G}}\left(\overline{\mathsf{sk}[\mathbf{y}]} = (\mathsf{sk}[\mathbf{y}], L_\mathsf{k}), \overline{\mathsf{ct}[\mathbf{x}]} = (\mathsf{ct}[\mathbf{x}], L_\mathsf{e}), \langle \mathbf{y}, \mathbf{x} \rangle\right) = 1 \mid \neg \mathsf{Bad}\right] \geqslant \nu$

Proof. We have to argue that $\overline{\mathrm{Dec}}^{\mathbb{G}}$ simulates the $\mathbb{L}$ oracle for $\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], \langle \mathbf{y}, \mathbf{x} \rangle)$ correctly and consistently with the $\mathbb{L}$ queries made during $\mathrm{KGen}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{y})$ and $\mathrm{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x})$. Consider a $\mathsf{PointerToLabel}(i)$ that Dec makes. If the group element $T(i)$'s corresponding label can be found in $L_\mathsf{e} \cup L_\mathsf{k}$, the query is answered with this label and it is clear that this answer faithfully simulates the oracle. If such a group element is not found, a new label Label is sampled uniformly at random (excluding all known labels for different group elements). That a label for this group element was queried before during the Enc or KGen call is excluded by the event $\mathsf{Intersection_e}$ or $\mathsf{Intersection_k}$. These two events also exclude that a $\mathsf{LabelToPointer}$ query that was answered with this group element before was made. Thus, a faithful simulation of this query should be answered with a random label that is uniformly random among all unused labels. That the simulated label has been used before is captured by the $\mathsf{CollL_e}$ and $\mathsf{CollL_k}$ events. The argument that the simulation of $\mathsf{LabelToPointer}$ queries is faithful is analogous. □

Next, we need to bound the probability of each of the bad events defined above occurs.

Claim C.6 For every polynomial poly there exists a polynomial poly' such that by setting $\ell_\mathsf{e} := \mathsf{poly}'(\kappa)$ we have $\Pr[\mathsf{Intersection_e}] \leqslant 1/\mathsf{poly}(\kappa)$.

Proof. For this argument, fix all random variables except $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell_\mathsf{e}}$ and V (i.e. fix $\mathbf{y}$, the random coins used by all the algorithms (except those that correspond to sampling $\mathbf{x}_1, \ldots, \mathbf{x}_{\ell_\mathsf{e}}$) and the oracles). Moreover, we assume that $\mathbf{x}_1, \ldots, \mathbf{x}_{\ell_\mathsf{e}}$ contains no duplicate vector. This can happen only with negligible probability, because $|\mathsf{V}| = q^{n-1}$ is of superpolynomial size.

We first show $\Pr[\mathsf{Intersection_e} \mid \mathbf{x} \in \mathsf{V}] \leqslant 2/(\mathsf{e} \cdot \ell_\mathsf{e})$. Fix $i \in [q_\mathsf{d}]$, where q_d is the number of $\mathbb{L}$ queries that Dec makes and let t_i be the probability that the i-th $\mathbb{L}$ query made during $\mathrm{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{ct}[\mathbf{x}], \mathsf{sk}[\mathbf{y}], \langle \mathbf{x}, \mathbf{y} \rangle)$ is also made during $\mathsf{ct}[\mathbf{x}] \leftarrow \mathrm{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x})$. Here the probability is taken over $\mathbf{x} \leftarrow \mathsf{V}$ and all other variables are fixed. The probability that this query causes the $\mathsf{Intersection_e}$ is at most $t_i \cdot (1 - t_i)^{\ell_\mathsf{e}} \leqslant 2/(e \cdot \ell_\mathsf{e})$, where e is Euler's constant.

Next, we show $\Pr[\mathsf{Intersection_e} \mid \mathbf{x} \notin \mathsf{V}] \leqslant 1/\mathsf{poly}(\kappa)$ using Lemma F.1. Let $F(\mathbf{x} \in \mathbb{Z}_q^n)$ be the set of $\mathbb{L}$ queries made both during $\mathsf{ct}[\mathbf{x}] \leftarrow \mathrm{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x})$ and $\mathrm{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{ct}[\mathbf{x}], \mathsf{sk}[\mathbf{y}], \langle \mathbf{x}, \mathbf{y} \rangle)$. Note that F is a function, because $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell_\mathsf{e}}$ are disjoint (by assumption) and we have fixed the oracles and the randomness for each call of an algorithm.

$\mathsf{Intersection_e}$ occurs here iff $F(\mathbf{x}) \nsubseteq \bigcup_{i=1}^{\ell_\mathsf{e}} F(\mathbf{x}_i)$. Lemma F.1 guarantees that for every polynomial poly there exists a polynomial poly' such that by setting $\ell_\mathsf{e} := \mathsf{poly}'(\kappa)$ this even occurs at most with probability $1/\mathsf{poly}(\kappa)$. Note that the marginal distribution of V, as sampled by the compiler, is uniformly random and thus we can apply Lemma F.1 here. □

Claim C.7 For every polynomial poly there exists a polynomial poly' such that by setting $\ell_{\mathsf{e}} := \mathsf{poly}'(\kappa)$ we have $\Pr[\mathsf{Intersection}_{\mathsf{k}}] \leqslant 1/\mathsf{poly}(\kappa)$.

The proof of this claim is analogous to the proof of Claim C.6. The only difference is that we have to use Lemma F.2 instead of Lemma F.1, because U is an $n-2$ dimensional subspace.

Claim C.8 $\Pr[\mathsf{CollG}_{\mathsf{e}}] \leqslant q_{\mathsf{d},\mathsf{i}} q_{\mathsf{e},\mathsf{i}}/p$, $\Pr[\mathsf{CollG}_{\mathsf{k}}] \leqslant q_{\mathsf{d},\mathsf{i}} q_{\mathsf{k},\mathsf{i}}/p$, $\Pr[\mathsf{CollL}_{\mathsf{e}}] \leqslant q_{\mathsf{d},\mathsf{l}} q_{\mathsf{e},\mathsf{l}}/|\mathsf{LB}|$, and $\Pr[\mathsf{CollL}_{\mathsf{k}}] \leqslant q_{\mathsf{d},\mathsf{l}} q_{\mathsf{k},\mathsf{l}}/|\mathsf{LB}|$, where $q_{\mathsf{e},\mathsf{i}}$, $q_{\mathsf{k},\mathsf{i}}$, and $q_{\mathsf{d},\mathsf{i}}$ are the maximum number of $\mathsf{LabelToPointer}$ queries made during Enc, KGen and Dec, respectively, and $q_{\mathsf{e},\mathsf{l}}$, $q_{\mathsf{k},\mathsf{l}}$, and $q_{\mathsf{d},\mathsf{l}}$ are the maximum number of $\mathsf{PointerToLabel}$ queries made during Enc, KGen and Dec.

Proof. Each of these events can only occur if a randomly sampled label or group element exponent in a simulated $\mathbb{L}$ query made during $\mathrm{Dec}^{\mathbb{G},\mathbb{L}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}])$ was already made during $\mathrm{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x}$ or $\mathrm{KGen}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{y})$, respectively. Since these labels and exponents are sampled uniformly at random from $\mathbb{Z}_p$ or LB, the claim above follows. □

Combining Claims C.5 to C.8 shows correctness of the compiled scheme.

Key hiding. We give a reduction that reduces an adversary breaking the (U,V)-key hiding security of the compiled scheme $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$, for the subspaces U and V selected during $\overline{\mathrm{Setup}}(1^\kappa)$ of the compiled scheme to an adversary breaking the (U,V)-key hiding security of the underlying IPFE $\overline{\mathcal{E}}$.

Initially, the reduction makes ℓ_{e} encryption queries to receive $(\mathbf{v}_1, \mathsf{ct}_1), \ldots, (\mathbf{v}_{\ell_{\mathsf{e}}}, \mathsf{ct}_{\ell_{\mathsf{e}}})$ and ℓ_{k} straight key corruption queries to receive $(\mathbf{u}_1, \mathsf{sk}_1), \ldots, (\mathbf{u}_{\ell_{\mathsf{k}}}, \mathsf{sk}_{\ell_{\mathsf{k}}})$. If the adversary makes a straight key corruption query, the reduction forwards this query to receive $(\mathbf{u}, \mathsf{sk}[\mathbf{u}])$. It then runs steps 2 and 3 of the $\overline{\mathrm{KGen}}^{\mathbb{G},\mathbb{L}}$ algorithm of the compiler with the plaintext/ciphertext pairs $(\mathbf{v}_1, \mathsf{ct}_1), \ldots, (\mathbf{v}_{\ell_{\mathsf{e}}}, \mathsf{ct}_{\ell_{\mathsf{e}}})$ obtained in the first step to compute L_{k}. It returns $(\mathbf{u}, \overline{\mathsf{sk}}[\mathbf{u}] = (\mathsf{sk}[\mathbf{u}], L_{\mathsf{k}}))$ to the adversary.

If the adversary makes a LoR key corruption query, the reduction forwards this query to receive $(\mathbf{w}, \mathbf{w}', \mathsf{sk})$ and sends $(\mathbf{w}, \mathbf{w}', \overline{\mathsf{sk}} = (\mathsf{sk}, L_{\mathsf{k}}))$ to the adversary, where L_{k} is computed as for straight key corruption queries.

If the adversary makes an encryption query, the reduction forwards this query to receive $(\mathbf{v}, \mathsf{ct}[\mathbf{v}])$. It then runs steps 2 and 3 of the $\overline{\mathrm{Enc}}^{\mathbb{G},\mathbb{L}}$ algorithm of the compiler with the vector/key pairs $(\mathbf{u}_1, \mathsf{sk}_1), \ldots, (\mathbf{u}_{\ell_{\mathsf{k}}}, \mathsf{sk}_{\ell_{\mathsf{e}}})$ obtained in the first step to compute L_{e}. It returns $(\mathbf{v}, \overline{\mathsf{ct}}[\mathbf{v}] = (\mathsf{sk}[\mathbf{v}], L_{\mathsf{e}}))$ to the adversary.

All $\mathbb{G}$ and $\mathbb{L}$ queries of the adversary are answered by forwarding them to the $\mathbb{G}$ or $\mathbb{L}$ oracle of the reduction. The reduction returns the same challenge bit as the adversary. Clearly, the reduction simulates the (U,V)-key hiding game of the compiled IPFE $\mathcal{E}^{\mathbb{G},\mathbb{L}}$ perfectly.

Plaintext hiding. We give a reduction that is analogous the reduction for key hiding and reduces an adversary breaking the (U,V)-plaintext hiding security of the compiled scheme $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$, for the subspaces U and V selected during $\overline{\mathrm{Setup}}(1^\kappa)$ of the compiled scheme to an adversary breaking the (U,V)-plaintext hiding security of the underlying IPFE $\overline{\mathcal{E}}$.

Initially, the reduction makes ℓ_{e} encryption queries to receive $(\mathbf{v}_1, \mathsf{ct}_1), \ldots, (\mathbf{v}_{\ell_{\mathsf{e}}}, \mathsf{ct}_{\ell_{\mathsf{e}}})$ and ℓ_{k} straight key corruption queries to receive $(\mathbf{u}_1, \mathsf{sk}_1), \ldots, (\mathbf{u}_{\ell_{\mathsf{k}}}, \mathsf{sk}_{\ell_{\mathsf{k}}})$. If the adversary makes a key corruption query, the reduction forwards this query to receive $(\mathbf{u}, \mathsf{sk}[\mathbf{u}])$. It then runs steps 2 and 3 of the $\overline{\mathrm{KGen}}^{\mathbb{G},\mathbb{L}}$ algorithm of the compiler with the plaintext/ciphertext pairs $(\mathbf{v}_1, \mathsf{ct}_1), \ldots, (\mathbf{v}_{\ell_{\mathsf{e}}}, \mathsf{ct}_{\ell_{\mathsf{e}}})$ obtained in the first step to compute L_{k}. It returns $(\mathbf{u}, \overline{\mathsf{sk}}[\mathbf{u}] = (\mathsf{sk}[\mathbf{u}], L_{\mathsf{k}}))$ to the adversary.

If the adversary makes the special key corruption query, the reduction forwards this query to receive $(\mathbf{y}, \mathsf{sk}[\mathbf{y}])$ and sends $(\mathbf{y}, \overline{\mathsf{sk}}[\mathbf{y}] = (\mathsf{sk}[\mathbf{y}], L_{\mathsf{k}}))$ to the adversary, where L_{k} is computed as for regular key corruption queries.

If the adversary makes a straight encryption query, the reduction forwards this query to receive $(\mathbf{v}, \mathsf{ct}[\mathbf{v}])$. It then runs steps 2 and 3 of the $\overline{\mathrm{Enc}}^{\mathbb{G},\mathbb{L}}$ algorithm of the compiler with the vector/key pairs $(\mathbf{u}_1, \mathsf{sk}_1), \ldots, (\mathbf{u}_{\ell_{\mathsf{k}}}, \mathsf{sk}_{\ell_{\mathsf{e}}})$ obtained in the first step to compute L_{e}. It returns $(\mathbf{v}, \overline{\mathsf{ct}}[\mathbf{v}] = (\mathsf{sk}[\mathbf{v}], L_{\mathsf{e}}))$ to the adversary.

If the adversary makes a LoR encryption query, the reduction forwards this query to receive (z, ct) and sends $(z, \overline{\mathsf{ct}} = (\mathsf{ct}, L_{\mathsf{e}}))$ to the adversary, where L_{e} is computed as for straight encryption queries.

All $\mathbb{G}$ and $\mathbb{L}$ queries of the adversary are answered by forwarding them to the $\mathbb{G}$ or $\mathbb{L}$ oracle of the reduction. The reduction returns the same challenge bit as the adversary. Clearly, the reduction simulates the (U, V)-plaintext hiding game of the compiled IPFE $\mathcal{E}^{\mathbb{G},\mathbb{L}}$ perfectly.

C.3 Step 2: Replacing label-queries in KGen and Enc with a random permutation

For the purpose of the next compilation, we introduce the following oracle $\mathcal{R}$ which is defined with respect to an injective label function $\mathsf{R}\colon \mathbb{Z}_p \mapsto \mathsf{LB}$. The oracle $\mathcal{R}$ gives the adversary access to the following two suboracles.

- **Evaluate queries EvalR:** A $\mathsf{EvalR}(x \in \mathbb{Z}_p)$ query is answered with $\mathsf{R}(x)$
- **Inversion queries InvR:** A $\mathsf{InvR}(\mathsf{Label} \in \mathsf{LB})$ query is answered with x if there exists an $x \in \mathbb{Z}_p$ with $\mathsf{R}(x) = \mathsf{Label}$ and with $\perp$ otherwise.

The function $\mathsf{R}\colon \mathbb{Z}_p \mapsto \mathsf{LB}$ is chosen initially uniformly at random among the functions $\mathsf{R}\colon \mathbb{Z}_p \mapsto \mathsf{LB}$, independently of all other oracles.

The above oracle (and, in fact, a keyed version of it[12]) can be built from a random oracle as follows. Use the random oracle to build a pseudorandom permutation (PRP) [32] with domain $[0, |\mathsf{LB}| - 1]$ and co-domain LB. The $\mathsf{EvalR}(x)$ oracle is then implemented by evaluating the PRP on x. The $\mathsf{InvR}(\mathsf{Label})$ oracle is implemented by evaluating the inverse PRP on Label and replacing all results $x \notin [0, p-1]$ with $\perp$.

Lemma C.9 *Assume $\mathcal{E}^{\mathbb{G},\mathbb{L}} := (\mathrm{Setup}^{\mathbb{G},\mathbb{L}}, \mathrm{KGen}^{\mathbb{G},\mathbb{L}}, \mathrm{Enc}^{\mathbb{G},\mathbb{L}}, \mathrm{Dec}^{\mathbb{G}})$ is an IPFE for $\mathbb{Z}_q^n$ where* Dec *makes no $\mathbb{L}$ queries.*

Assume that $\mathcal{E}^{\mathbb{G},\mathbb{L}}$ is ν-correct and that $\mathcal{E}^{\mathbb{G},\mathbb{L}}$ is both (U, V)-key hiding (Definition C.2) and (U, V)-plaintext hiding (Definition C.3) for subspaces $\mathsf{U}, \mathsf{V} \subseteq \mathbb{Z}_q^n$ that satisfy Definition C.1 (these subspaces may be chosen by the IPFE scheme).

Then the compiled scheme $\overline{\mathcal{E}}^{\mathbb{G},\mathcal{R}} := (\overline{\mathrm{Setup}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{KGen}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{Enc}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{Dec}}^{\mathbb{G}})$ defined in Figure 4, where $\mathcal{R}$ is the oracle defined above, continue to be PPT algorithms, and

1. *$\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$ is ν-correct,*
2. *and for the same subspaces $\mathsf{U}, \mathsf{V} \subseteq \mathbb{Z}_q^n$ it is*
   (a) *$\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$ is (U, V)-key hiding and*
   (b) *$\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$ is (U, V)-plaintext hiding*

   against adversaries $\mathcal{A}^{\mathbb{G},\mathbb{L}}$ that continue to have oracle access to $\mathbb{G}, \mathbb{L}$ but do not have access to the oracle $\mathcal{R}$.

[12] The reason we do not need a keyed version for our compiler is that we do not give the adversary access to the oracle $\mathcal{R}$.

$\overline{\text{Setup}}^{\mathbb{G},\mathcal{R}}(1^\kappa)$:

1. Sample msk according to $\text{Setup}^{\mathbb{G},\mathbb{L}}(1^\kappa)$ and simulate the oracle $\mathbb{L}$ as follows:
   (a) Keep track of the discrete logarithm of every group element created in a table D. This is possible because $\text{Setup}^{\mathbb{G},\mathbb{L}}$ gets no group elements as input.
   (b) Simulate $\mathsf{PointerToLabel}(x \cdot g)$ queries by returning $\mathsf{EvalR}(x)$. The discrete log x of $x \cdot g$ can be looked up in the table D.
   (c) Simulate $\mathsf{LabelToPointer}(\mathsf{Label})$ queries by running $x \leftarrow \mathsf{InvR}(\mathsf{Label})$ and returning $x \cdot g$.

Let the resulting master secret key $\overline{\mathsf{msk}}$ contain msk and D.

$\overline{\text{KGen}}^{\mathbb{G},\mathcal{R}}(\overline{\mathsf{msk}} := (\mathsf{msk}, D), \mathbf{y})$**:**

1. Sample $\mathsf{sk}[\mathbf{y}] \leftarrow \text{KGen}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{y})$ and simulate the oracle $\mathbb{L}$ as follows:
   (a) Keep track of the discrete logarithm of every group element created in the table D. This is possible because the table initially already contains the discrete log of every input group element.
   (b) Simulate $\mathsf{PointerToLabel}(x \cdot g)$ queries by returning $\mathsf{EvalR}(x)$. The discrete log x of $x \cdot g$ can be looked up in the table D.
   (c) Simulate $\mathsf{LabelToPointer}(\mathsf{Label})$ queries by running $x \leftarrow \mathsf{InvR}(\mathsf{Label})$ and returning $x \cdot g$.
2. Output $\mathsf{sk}[\mathbf{y}]$.

$\overline{\text{Enc}}^{\mathbb{G},\mathcal{R}}(\overline{\mathsf{msk}} := (\mathsf{msk}, D), \mathbf{x})$**:**

1. Sample $\mathsf{ct}[\mathbf{x}] \leftarrow \text{Enc}^{\mathbb{G},\mathbb{L}}(\mathsf{msk}, \mathbf{x})$ and simulate the oracle $\mathbb{L}$ as follows:
   (a) Keep track of the discrete logarithm of every group element created in the table D. This is possible because the table initially already contains the discrete log of every input group element.
   (b) Simulate $\mathsf{PointerToLabel}(x \cdot g)$ queries by returning $\mathsf{EvalR}(x)$. The discrete log x of $x \cdot g$ can be looked up in the table D.
   (c) Simulate $\mathsf{LabelToPointer}(\mathsf{Label})$ queries by running $x \leftarrow \mathsf{InvR}(\mathsf{Label})$ and returning $x \cdot g$.
2. Output $\mathsf{ct}[\mathbf{x}]$.

$\overline{\text{Dec}}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], z)$**:** Output $\text{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], z)$

Figure 4: The compiler for Lemma C.9 that compiles an IPFE $\mathcal{E}^{\mathbb{G},\mathbb{L}} := (\text{Setup}^{\mathbb{G},\mathbb{L}}, \text{KGen}^{\mathbb{G},\mathbb{L}}, \text{Enc}^{\mathbb{G},\mathbb{L}}, \text{Dec}^{\mathbb{G}})$ into an IPFE $\overline{\mathcal{E}}^{\mathbb{G},\mathcal{R}} := (\overline{\text{Setup}}^{\mathbb{G},\mathcal{R}}, \overline{\text{KGen}}^{\mathbb{G},\mathcal{R}}, \overline{\text{Enc}}^{\mathbb{G},\mathcal{R}}, \overline{\text{Dec}}^{\mathbb{G}})$.

Proof of Lemma C.9.

Correctness. Let $\overline{\mathsf{msk}} = (\mathsf{msk}, D) \leftarrow \overline{\mathrm{Setup}}^{\mathbb{G},\mathcal{R}}(1^\kappa)$, $\mathbf{x} \leftarrow \mathbb{Z}_q^n$, $\mathbf{y} \leftarrow \mathbb{Z}_q^n$, $\mathsf{sk}[\mathbf{y}] \leftarrow \overline{\mathrm{KGen}}^{\mathbb{G},\mathcal{R}}(\overline{\mathsf{msk}}, \mathbf{y})$, and $\mathsf{ct}[\mathbf{x}] \leftarrow \overline{\mathrm{Enc}}^{\mathbb{G},\mathcal{R}}(\overline{\mathsf{msk}}, \mathbf{x})$. The simulated oracle answers for the $\mathbb{L}$ oracle using the $\mathcal{R}$ oracle are identically distributed to the answers produced by the real $\mathbb{L}$ oracle. Thus, the marginal distribution of $(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}])$ is identical to the marginal distribution of the ciphertext/key generated by the original IPFE scheme. Since Dec has no oracle access to $\mathbb{L}$, the output distribution of Dec is identical when fed with $\mathsf{sk}[\mathbf{y}]$ and $\mathsf{ct}[\mathbf{x}]$) compared to a ciphertext/key generated with the real oracle $\mathbb{L}$, and we get $\Pr\left[\overline{\mathrm{Dec}}^{\mathbb{G}}(\mathsf{sk}[\mathbf{y}], \mathsf{ct}[\mathbf{x}], \langle \mathbf{y}, \mathbf{x} \rangle) = 1\right] \geqslant \nu$.

Key hiding. We give a reduction that reduces an adversary breaking the (U, V)-key hiding security of the compiled scheme $\overline{\mathcal{E}}^{\mathbb{G},\mathbb{L}}$, for the subspaces U and V selected during $\overline{\mathrm{Setup}}(1^\kappa)$ of the compiled scheme to an adversary breaking the (U, V)-key hiding security of the underlying IPFE $\overline{\mathcal{E}}$.

The reduction forwards all $\mathbb{G}$, straight and LoR key corruption queries and encryption queries to the underlying game and returns the respective answers of these oracles. Only $\mathbb{L}$ queries are not forwarded. Instead, the reduction simulates the answers to these queries similar to step 2 of the $\overline{\mathrm{Dec}}^{\mathbb{G}}$ algorithm in Lemma C.4. Concretely, the reduction

1. initializes a table $L_{\mathsf{d}} := \emptyset$.
2. $\mathsf{PointerToLabel}(g_1)$ **queries** are answered as follows. Search L_{d} for an entry of the form (g_1, Label). If such an entry is found, return Label. Otherwise, sample a new label $\mathsf{Label} \leftarrow \mathsf{LB} \setminus \{\mathsf{Label} \mid (_, \mathsf{Label}) \in L_{\mathsf{d}}\}$, add (g_1, Label) to L_{d} and return Label.
3. $\mathsf{LabelToPointer}(\mathsf{Label})$ **queries** are answered as follows. Search L_{d} for an entry of the form (g_1, Label). If such an entry is found, return g_1. Otherwise, with probability $1 - p/|\mathsf{LB}|$, store $(\bot, \mathsf{Label})$ in L_{d} and return $\bot$. In the remaining case, sample $x \leftarrow \mathbb{Z}_q \setminus \{z \mid (z \cdot g, _) \in L_{\mathsf{d}}\}$, return $x \cdot g$, and add $(x \cdot g, \mathsf{Label})$ to L_{d}.

The reduction thus simulates the oracle $\mathbb{L}$ for the adversary perfectly, but independently of the real oracle $\mathbb{L}$. This means the reduction simulates the (U, V)-key hiding game of the compiled IPFE $\mathcal{E}^{\mathbb{G},\mathbb{L}}$ perfectly, because the compiled IPFE is equivalent to running the original IPFE with an independently sampled $\mathbb{L}$ oracle.

The reduction returns the same challenge bit as the adversary and thus achieves the same advantage.

Plaintext hiding. The reduction that for the (U, V)-plaintext hiding security of the compiled scheme is completely analogous the reduction for key hiding security. It forwards all queries of the adversary except for $\mathbb{L}$ queries, which are simulated independently of the real $\mathbb{L}$ oracle, as described in the key hiding game. This perfectly simulates the (U, V)-plaintext hiding game for the adversary. The reduction returns the same challenge bit as the adversary and achieves the same advantage.

C.4 Step 3: Removing Maurer Group Query from KGen

For the third step of our proof, we will remove Maurer's query $\mathbb{G}$ in the KGen. The compiler is exactly the same as presented in Section 4.1, and we prove the following lemma.

Lemma C.10 *Assume* $(\mathrm{Setup}^{\mathbb{G},\mathcal{R}}, \mathrm{KGen}^{\mathbb{G},\mathcal{R}}, \mathrm{Enc}^{\mathbb{G},\mathcal{R}}, \mathrm{Dec}^{\mathbb{G}})$ *is* ν*-correct and satisfies both* (U, V)*-key hiding (Definition C.2) and* (U, V)*-plaintext hiding (Definition C.3) for every some subspace* U *and*

V. *The compiled scheme* $\overline{\mathcal{E}}^{\mathbb{G},\mathcal{R}} := (\overline{\mathrm{Setup}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{KGen}}^{\mathcal{R}}, \overline{\mathrm{Enc}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{Dec}}^{\mathbb{G}})$*, defined for the given* V*, has the property that* $\overline{\mathrm{KGen}}^{\mathcal{R}}$ *makes no* $\mathbb{G}$ *queries, and* $\overline{\mathrm{Setup}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{Enc}}^{\mathbb{G},\mathcal{R}}, \overline{\mathrm{Dec}}^{\mathbb{G}}$ *continue to be PPT algorithms, and*

1. $\overline{\mathcal{E}}^{\mathbb{G},\mathcal{R}}$ *is* $(\nu - 1/q - 1/\mathsf{poly}(\kappa))$*-correct for an arbitrarily chosen polynomial* $\mathsf{poly}(\kappa)$*.*
2. $\overline{\mathcal{E}}^{\mathbb{G},\mathcal{R}}$ *preserves* (U,V)*-key hiding (Definition C.2).*

The compilation proceeds in the exact same way. That is, it will sample many ciphertexts encrypting random vectors from the subspace V during $\overline{\mathrm{Setup}}^{\mathbb{G},\mathcal{R}}$ and store them in pp. $\overline{\mathrm{KGen}}^{\mathcal{R}}$ removes group elements from secret keys and replaces them with the set R of linear relations observed by decryption the ciphertexts in pp with the secret key. Now, instead of repeating the construction and the proof, we only comment on the minor difference below.

- All algorithm has access to not only $\mathbb{G}$, but also a random oracle $\mathcal{R}$. However, our compiler is oblivious to whether these algorithms have access to additional oracles or not. It simply removes KGen's access to $\mathbb{G}$ (and maintains the original algorithm's access/inaccess to $\mathcal{R}$).
- The fact that $\overline{\mathrm{KGen}}^{\mathcal{R}}$ makes no $\mathbb{G}$ query and that $\overline{\mathcal{E}}^{\mathbb{G},\mathcal{R}}$ preserves (U,V)-key hiding can be argued in the identical way as before.
- The argument for correctness will be slightly different. In particular, we argue correctness by claiming that when we try to decrypt a ciphertext encrypting a random vector from the entire space using linear relations learned by drawing ciphertexts from V. The probability that we run into a linear query outside the span of the relations that we learn is at most $m/(\ell+1) + 1/q + \mathrm{negl}(\kappa)$. Note that this implies the correctness drop is bounded by $m/(\ell+1) + 1/q + \mathrm{negl}(\kappa)$ (which can be made smaller than $1/q + 1/\mathsf{poly}(\kappa)$ for an arbitrarily chosen polynomial) by the same argument in the original analysis.
  We prove this claim via the (U,V)-plaintext property (Definition C.3). Suppose the probability that a linear query outside the span of the relations is asked is at least $m/(\ell+1) + 1/q + \varepsilon$, where ε is non-negligible. Then we break the (U,V)-plaintext property as follows.
  The adversary first makes one special key corruption query for a secret key sk. (We use $\mathbf{y}$ to denote sk in the argument below.) Note that this $\mathbf{y}$ is drawn conditioned on $\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\})^\perp \nsubseteq \mathsf{V}$. We first prove that, if sk is drawn uniformly randomly, then this condition fails with probability $\leqslant 1/q$. This is because (note that U and V has dimension $n-2$ and $n-1$, respectively)

$$\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\})^\perp \subseteq \mathsf{V} \iff \mathsf{V}^\perp \subseteq \mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}) \iff \mathbf{y} \in \mathsf{Span}(\mathsf{U} \cup \mathsf{V}^\perp).$$

  Since (U,V) is a valid pair (Definition C.1), $\mathsf{V}^\perp \nsubseteq \mathsf{U}$, and thus $\dim \mathsf{Span}(\mathsf{U} \cup \mathsf{V}^\perp) = n-1$, so the bad set has size q^{n-1} and

$$\Pr[\text{bad}] = \frac{q^{n-1}}{q^n} = \frac{1}{q}, \quad \text{hence} \quad \Pr[\text{requirement holds}] = 1 - \frac{1}{q}.$$

  This completes the argument. Note that this implies that the distribution of sk is at most $1/q$ far apart from uniformly random.
  Now, we make ℓ straight encryption queries to get ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ (and corresponding plaintexts) from the space V. It learns the relations R by decrypting $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ with sk. Next, it makes one LoR encryption query to get one ciphertext ct', which encrypt either vectors from V or the entire space. Then we decrypt and test if a linear query outside the span of

the relation is queried. If such a query is not asked, the adversary guesses the bit b to be 0. Otherwise, the adversary guesses the bit b to be 1.
This breaks (U, V)-plaintext property because: (i) On the one hand, if $b = 0$, then by Lemma 4.3, the probability that linear query outside the span will be asked is upper bounded by $m/(\ell+1)$ (this is because both the new ciphertext ct and the ciphertexts in pp are coming from V). (ii) On the other hand, if $b = 1$, then by our assumption, when sk is uniformly random, a linear query outside the span will be asked with probability $\geqslant m/(\ell+1) + 1/q + \varepsilon$. And we know that, the sk is drawn according to a distribution that is at most $1/q$ far apart from a uniformly random distribution. Therefore, this means that a linear query outside the span will be asked for this sk with probability at least $m/(\ell+1) + \varepsilon$. Therefore, it gives the adversary an advantage of ε, which is non-negligible. This concludes the proof of correctness and the entire Lemma C.10.

C.5 Step 4: Information-Theoretic Attack

We first begin with some notation. Let $\mathcal{E}^{\mathbb{G},\mathcal{R}} := (\mathrm{Setup}^{\mathbb{G},\mathcal{R}}, \mathrm{KGen}^{\mathcal{R}}, \mathrm{Enc}^{\mathbb{G},\mathcal{R}}, \mathrm{Dec}^{\mathbb{G}})$ be an IPFE, where $\mathbb{G}$ is a Maurer GGM, and $\mathcal{R}$ is an arbitrary oracle (e.g., RO), and where KGen makes no $\mathbb{G}$ queries and Dec makes no $\mathcal{R}$ queries. We use the word 'Limited' for such IPFE constructions in this section.

Let $\mathsf{msk} \in \mathrm{Setup}^{\mathbb{G},\mathcal{R}}(r_{\mathsf{setup}})$, with Q_{setup} denoting the underlying set of Q-A pairs. Let $\mathsf{sk}[\mathbf{u}] \in \mathrm{KGen}^{\mathcal{R}}(\mathsf{msk}, \mathbf{u})$ for some vector $\mathbf{u}$. Let $\mathsf{ct}[\mathbf{x}] \in \mathrm{Enc}^{\mathbb{G},\mathcal{R}}(\mathsf{msk}, \mathbf{x}; r_{\mathsf{enc}})$ for some plaintext vector $\mathbf{x}$, with Q_{enc} denoting the underlying set of Q-A pairs.

With the above parameters, $\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{x}], z := \langle \mathbf{u}, \mathbf{x} \rangle)$ outputs 1 with some probability α (e.g., $\alpha = 1$ if the scheme is perfectly correct). Recall that by Definition 3.5 $\mathrm{Dec}^{\mathbb{G}} = (\mathrm{Dec}_0^{\mathbb{G}}, \mathrm{Dec}_1^{\mathbb{G}})$, where $\mathrm{Dec}_0^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{x}])$ outputs a group element $\widetilde{g}$, and $\mathrm{Dec}_1^{\mathbb{G}}$ outputs the result of an equality query on $(\widetilde{g}, z \cdot g)$, where g is the generator. Now if we have all the underlying parameters (r_{setup}, $Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}$), we can compute the discrete log of $\tilde{g}$, the output of $\mathrm{Dec}_0^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{x}])$ **without making any $\mathbb{G}$ queries**. This is because $\mathrm{KGen}^{\mathcal{R}}$, which produces $\mathsf{sk}[\mathbf{u}]$, makes no $\mathbb{G}$ queries; hence, we possess the discrete logs of all group elements generated so far. Consequently, there exists an algorithm $\mathrm{DecFull}(\mathsf{sk}[\mathbf{u}], (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}))$ which outputs an element $f \in \mathbb{Z}_q$, which equals $\langle \mathbf{u}, \mathbf{x} \rangle$ with the same probability α as above. Concretely, the algorithm DecFull, on the above input, can simulate the execution of $\mathrm{Dec}_0^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{x}])$, without making any $\mathbb{G}$ queries, and extract and output the discrete log of the group element $\tilde{g}$ outputted by Dec_0. Our attacker will make use of the above (oracle-free) algorithm DecFull. Because our attacker will perform a brute-force search, it may invoke DecFull on inputs that are not 'valid transcripts'; therefore, we give the full description of DecFull. We start with the following terminology.

Definition C.11 We say a set Q of Q-A pairs is minimal for an oracle-based execution, say, $\mathrm{Enc}^{Q}(\mathsf{msk}, \mathbf{x}; r)$ if all the queries asked during the execution are answered in Q and if every query in Q is asked by the execution.

We now define $\mathrm{DecFull}(\mathsf{sk}[\mathbf{u}], (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}))$.

1. Consistency and validity of Q-A pairs: run $\mathsf{msk} := \mathrm{Setup}^{Q_{\mathsf{setup}}}(1^{\kappa}; r_{\mathsf{setup}})$ and output $\perp$ if Q_{setup} is not a minimal set of Q-A pairs for this execution. Next, run $\mathsf{ct}[\mathbf{x}] := \mathrm{Enc}^{Q_{\mathsf{enc}}}(\mathsf{msk}, \mathbf{x}; r_{\mathsf{enc}})$ and, likewise, output $\perp$ if Q_{enc} is not a minimal set of Q-A pairs for this execution.

2. Output $\perp$ if $Q_{\mathsf{setup}} \cup Q_{\mathsf{enc}}$ is inconsistent.[13]
3. Simulate the execution of $\mathrm{Dec}_0^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{x}])$ without making any $\mathbb{G}$ queries and output the discrete log of the group element outputted by Dec_0.

We now describe the attacker and analyze the attacker's success probability.

Lemma C.12 (Breaking (U, V)-key hiding of limited IPFEs). *Suppose* $\mathcal{E}^{\mathbb{G},\mathcal{R}} := (\mathrm{Setup}^{\mathbb{G},\mathcal{R}}, \mathrm{KGen}^{\mathcal{R}}, \mathrm{Enc}^{\mathbb{G},\mathcal{R}}, \mathrm{Dec}^{\mathbb{G}})$ *is an IPFE over* $\mathbb{Z}_q^n$*, where for some polynomial* $\mathsf{poly} = \mathsf{poly}(\kappa)$

$$\Pr[\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{y}], \langle \mathbf{u}, \mathbf{y} \rangle) = 1] \geqslant \frac{2}{q} + \frac{1}{\mathsf{poly}}, \tag{21}$$

where the probability is taken over the random choice of $(\mathbb{G}, \mathcal{R})$*, and over* $\mathbf{u}, \mathbf{y} \leftarrow \mathbb{Z}_q^n$*,* $\mathsf{msk} \leftarrow \mathrm{Setup}^{\mathbb{G},\mathcal{R}}(1^\kappa)$*,* $\mathsf{sk}[\mathbf{u}] \leftarrow \mathrm{KGen}^{\mathcal{R}}(\mathsf{msk}, \mathbf{u})$ *and* $\mathsf{ct}[\mathbf{y}] \leftarrow \mathrm{Enc}^{\mathbb{G},\mathcal{R}}(\mathsf{msk}, \mathbf{y})$*. Then, there exists an adversary* $\mathcal{A}$ *that makes no queries and wins the* (U, V)*-key hiding game for any valid pair* (U, V) *with probability* $\frac{1}{2} + \frac{1}{2\mathsf{poly}} - \mathrm{negl}(\kappa)$*.*

Proof. We give the description of the attacker $\mathcal{A}$. Let ρ denote the total number of bits required to represent the bit-length of the randomness r_{setup} to $\mathrm{Setup}^{\mathbb{G},\mathcal{R}}(1^\kappa)$, the bit length of the set Q_{setup} of all Q-A pairs made during an execution of $\mathrm{Setup}^{\mathbb{G},\mathcal{R}}(1^\kappa)$, a vector $\mathbf{x}$ in $\mathbb{Z}_q^n$, the bit-length of the encryption randomness r_{enc}, and the bit-size of the set Q_{enc} of all Q-A pairs generated during an execution of Enc. Let $m = 64(\kappa + \rho)\mathsf{poly}^2$.

The adversary $\mathcal{A}$ makes m key-corruption queries to receive $(\mathbf{w}_1, \mathbf{u_1} := \mathbf{w}_1 + \mathbf{v}_1, \mathsf{sk}_1), \ldots$ $(\mathbf{w}_m, \mathbf{u}_m := \mathbf{w}_m + \mathbf{v}_m, \mathsf{sk}_m)$. Recall that for all $i \in [m]$: $\mathbf{w}_i \leftarrow \mathbb{Z}_q^n$, $\mathbf{v}_i \leftarrow \mathsf{V}^\perp$ and sk_i is a secret key for $\mathbf{w}_i + b\mathbf{v}_i$. $\mathcal{A}$ works based on a procedure Recover (defined momentarily) as follows.

1. Recalling ρ from above, if there exists $s \in \{0,1\}^\rho$, representing $(r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}})$, such that $\mathbf{x} \notin \mathbf{V}$ and that for at least $m(\frac{1}{q} + \frac{1}{4p})$ coordinates i, $f_i = \langle \mathbf{w}_i, \mathbf{x} \rangle$, where $(f_1, \ldots, f_m) = \mathrm{Recover}(r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$, return 0; else, return 1.

The procedure $\mathrm{Recover}(r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$ decrypts the ciphertext created by $(\mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}})$ with each sk_i and returns the results. Specifically, it does the following:

1. For $i \in [m]$, set $f_i := \mathrm{DecFull}(\mathsf{sk}_i, (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}))$, and output $(f_1, \ldots, f_m)$.

The proof of Lemma C.12 follows immediately from the following two lemmas. In the following b and b' denote, respectively, the challenge bit and $\mathcal{A}$'s output bit. □

Lemma C.13 $\Pr[b' = 0 \mid b = 0] \geqslant \frac{1}{2\mathsf{poly}} - \mathrm{negl}(\kappa)$.

Lemma C.14 $\Pr[b' = 0 \mid b = 1] \leqslant \mathrm{negl}(\kappa)$.

C.5.1 Proof of Lemma C.13.

Fix U and V. Let $\mathbb{M}$ be the space of all oracles $\mathcal{R}$; e.g., if $\mathcal{R}$ is a length preserving RO oracle, mapping t-bits to t-bits for all values of t, then $\mathbb{M}$ will be the set of all length-preserving functions.

We say a tuple $\eta := (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}})$ is good if it satisfies both conditions below.

[13] This condition may be triggered because our attacker, described later, performs a brute-force search and may invoke DecFull on all possible inputs, including invalid ones.

1. Well-formedness: for some oracle $\mathcal{R}' \in \mathbb{M}$, the sets Q_{setup} and Q_{Enc} correspond to the Q-A pairs made during the executions of, respectively, $\mathsf{msk} := \mathrm{Setup}^{\mathbb{G},\mathcal{R}'}(1^\kappa; r_{\mathsf{setup}})$ and $\mathsf{ct}[\mathbf{x}] := \mathrm{Enc}^{\mathbb{G},\mathcal{R}'}(\mathsf{msk}, \mathbf{x}; r_{\mathsf{enc}})$.[14] Moreover, we must have $\mathbf{x} \notin \mathbf{V}$.[15]
2. Correctness: The IPFE's correctness probability is at least $\frac{1}{q} + \frac{1}{2p}$ when all the above parameters are fixed and the rest are sampled randomly: sampling $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk}[\mathbf{u}] \leftarrow \mathrm{KGen}^{\mathcal{R}|\eta}(\mathsf{msk}, \mathbf{u})$, we have $\Pr[\mathrm{Dec}^{\mathbb{G}}(\mathsf{sk}[\mathbf{u}], \mathsf{ct}[\mathbf{x}], \langle \mathbf{u}, \mathbf{x} \rangle) = 1] \geqslant \frac{1}{q} + \frac{1}{2p}$. Here $\mathcal{R} \mid \eta$ denotes sampling $\mathcal{R}$ conditioned on being consistent with η — more precisely, consistent with the Q-A pairs in η.

Claim C.15 We have

$$\Pr[\eta := (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}) \text{ is } \mathsf{good}] \geqslant \frac{1}{2p},$$

where the probability is taken over $\mathcal{R}$, over $r_{\mathsf{setup}}, r_{\mathsf{enc}} \leftarrow \{0,1\}^*$, $\mathsf{msk} := \mathrm{Setup}^{\mathbb{G},\mathcal{R}}(1^\kappa; r_{\mathsf{setup}})$ with Q_{setup} being the underlying set of Q-A pairs, $\mathbf{x} \leftarrow \mathbb{Z}_q^n$, $\mathsf{ct}[\mathbf{x}] := \mathrm{Enc}^{\mathbb{G},\mathcal{R}}(\mathsf{msk}, \mathbf{x}; r_{\mathsf{enc}})$ with Q_{enc} being the underlying set of Q-A pairs.

Claim C.16 Fix a good tuple $\eta := (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}})$ generated as in Claim C.15. For all $i \in [m]$ sample $\mathbf{h}_i \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk}_i \leftarrow \mathrm{KGen}^{\mathcal{R}|\eta}(\mathsf{msk}, \mathbf{h}_i)$. With all but negligible probability over the choice of $(\mathbf{h}_i, \mathsf{sk}_i)$ pairs, $\mathrm{Recover}(r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$ agrees with $(\langle \mathbf{h}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{h}_m, \mathbf{x} \rangle)$ on at least $m(\frac{1}{q} + \frac{1}{4p})$ coordinates.

Proof of Lemma C.13 via Claims C.15 and C.16. Let msk be the challenge master secret key picked by the challenger, and let r_{setup} and Q_{setup} denote the underlying randomness and Q-A pairs. Also, sample $\mathbf{x} \leftarrow \mathbb{Z}_q^n$, $r_{\mathsf{enc}} \leftarrow \mathbb{Z}_q^n$, $\mathsf{ct}[\mathbf{x}] \leftarrow \mathrm{Enc}^{\mathbb{G},\mathcal{R}}(\mathsf{msk}, \mathbf{x}; r_{\mathsf{enc}})$. Let Ev be the event that the resulting $\eta := (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathsf{msk}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}, \mathsf{ct}[\mathbf{x}])$ is good. We claim that conditioned on Ev, the adversary $\mathcal{A}$ will output 0 with probability $(1 - \mathrm{negl}(\kappa))$. This will prove the lemma because the probability of Ev is at least $\frac{1}{2p}$.

Thus, assume the event Ev holds. Since $b = 0$, each sk_i given to $\mathcal{A}$ is a secret key for $\mathbf{w}_i \leftarrow \mathbb{Z}_q^n$, sampled as $\mathsf{sk}_i \leftarrow \mathrm{KGen}^{\mathcal{R}|\eta}(\mathsf{msk}, \mathbf{w}_i)$. By Claim C.16, with probability $1 - \mathrm{negl}(\kappa)$, the output of $\mathrm{Recover}(r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$ agrees with $(\langle \mathbf{w}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{w}_m, \mathbf{x} \rangle)$ on at least $m(\frac{1}{q} + \frac{1}{4p})$ coordinates. Hence, the adversary $\mathcal{A}$ will output 0 with probability $1 - \mathrm{negl}(\kappa)$. This is because $\mathcal{A}$ goes over all possible values of $s \in \{0,1\}^\rho$ — to which η belongs (i.e., $\eta \in \{0,1\}^\rho$) — and outputs 0 as soon as for some s, the adversary gets at least $m(\frac{1}{q} + \frac{1}{4p})$ coordinate matches between the output of $\mathrm{Recover}(s, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$ and $(\langle \mathbf{w}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{w}_m, \mathbf{x} \rangle)$. The above already holds with all but negligible probability when setting $s = \eta$ □

Proof of Claim C.15. The proof follows via a simple averaging argument. Let α be the value of the probability in the claim and suppose to the contrary that $\alpha < \frac{1}{2p}$. Then, the probability of overall correctness is strictly less than $\frac{1}{2p} \cdot 1 + (\frac{1}{q} + \frac{1}{2p}) + \frac{1}{q}$, and hence strictly less than $\frac{2}{q} + \frac{1}{p}$, a contradiction. Here the term $\frac{1}{q}$ bounds the probability that the sampled $\mathbf{x}$ happens to be in V. □

[14] We only quantify over the oracle $\mathcal{R}'$ because the Maurer oracle $\mathbb{G}$ is deterministic and fixed.

[15] The requirement that $\mathbf{x} \notin \mathbf{V}$, which also appears in $\mathcal{A}$'s attack, is imposed for technical reasons and will be clarified in the proof of Lemma C.14.

Proof of Claim C.16. We prove this via the Chernoff bound. Fix a good tuple $\eta := (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}})$ as in the claim. Let

$$(f_1, \ldots, f_m) = \text{Recover}(\eta, \mathsf{sk}_1, \ldots, \mathsf{sk}_m).$$

Let msk and $\mathsf{ct}[\mathbf{x}]$ be the master secret key and ciphertext induced by the above fixed tuple η. Recall that for all i, $f_i = \text{DecFull}(\mathsf{sk}_i, (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}))$, where $\mathbf{h}_i \leftarrow \mathbb{Z}_q^n$ and $\mathsf{sk}_i \leftarrow \text{KGen}^{\mathcal{R}|\eta}(\mathsf{msk}, \mathbf{h}_i)$. For $i \in [m]$ let Ev_i be the event that $f_i = \langle \mathbf{h}_i, \mathbf{x} \rangle$. Since all these parameters, including those in η, are chosen honestly via the actual oracles, for all i, we have $f_i = \langle \mathbf{h}_i, \mathbf{x} \rangle$ iff $\text{Dec}^{\mathbb{G}}(\mathsf{sk}_i, \mathsf{ct}[\mathbf{x}], \langle \mathbf{h}_i, \mathbf{x} \rangle) = 1$. Thus, we may define the event Ev_i as $\text{Dec}^{\mathbb{G}}(\mathsf{sk}_i, \mathsf{ct}[\mathbf{x}], \langle \mathbf{h}_i, \mathbf{x} \rangle) = 1$.

We have

1. Ev_i's are i.i.d random variables.
2. $\mu := \Pr[\mathsf{Ev}_i = 1] \geqslant \frac{1}{q} + \frac{1}{2p}$. This follows by the tuple η being good.

Let $\mathsf{Ev} = \frac{\mathsf{Ev}_1 + \cdots + \mathsf{Ev}_m}{m}$. If $\mathsf{Ev} \geqslant \frac{1}{q} + \frac{1}{4p}$, then $\mathcal{A}$ will output 0. Let $\epsilon = \frac{1}{4p}$. Since $m = 64(\kappa + \rho)p^2$, we have $\frac{m\epsilon^2}{4} = (\kappa + \rho)$. Invoking the Chernoff bound (Lemma 3.1), the probability that $\mathcal{A}$ outputs 1 is at most

$$\Pr[\mathsf{Ev} < \frac{1}{q} + \frac{1}{4p}] \leqslant \Pr[|\mathsf{Ev} - \mu| \geqslant \frac{1}{4p}] \leqslant \frac{2}{e^{\frac{m\epsilon^2}{4}}} = \frac{2}{e^{\kappa+\rho}} = \text{negl}(\kappa), \tag{22}$$

as desired. □

C.5.2 Proof of Lemma C.14.

Proof. Suppose $b = 1$ as in the lemma. In this case for $\mathcal{A}$'s challenge input $(\mathbf{w}_1, \mathbf{u_1} := \mathbf{w}_1 + \mathbf{v}_1, \mathsf{sk}_1)$, $\ldots$ $(\mathbf{w}_m, \mathbf{u}_m := \mathbf{w}_m + \mathbf{v}_m, \mathsf{sk}_m)$ we have $\mathbf{w}_i \leftarrow \mathbb{Z}_q^n$, $\mathbf{v}_i \leftarrow \mathsf{V}^\perp$ and sk_i is a secret key for $\mathbf{u}_i$, for all $i \in [m]$.

We prove that for any fixed $\eta := (r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}})$, which involves ρ bits, the probability that this fixed η makes $\mathcal{A}$ output 0 is at most $\frac{2}{e^{\kappa+\rho}}$. The proof then follows by a union bound over all strings in $\{0,1\}^\rho$. Fix η as above. Let $(f_1, \ldots, f_m) = \text{Recover}(r_{\mathsf{setup}}, Q_{\mathsf{setup}}, \mathbf{x}, r_{\mathsf{enc}}, Q_{\mathsf{enc}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_m)$. Note that:

1. $(f_1, \ldots, f_m)$ only depends on $(\mathbf{u}_1, \ldots, \mathbf{u}_m)$, and is independent of $(\langle \mathbf{w}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{w}_m, \mathbf{x} \rangle)$. Here we make use of the fact that $x \notin \mathbf{V}$. Because if $x \in \mathbf{V}$, then for all i, $\langle \mathbf{w}_i, \mathbf{x} \rangle = \langle \mathbf{u}_i, \mathbf{x} \rangle$, and since $(f_1, \ldots, f_m)$ in turn depends on $(\mathbf{u}_1, \ldots, \mathbf{u}_m)$, we would not be able to say that $(\langle \mathbf{w}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{w}_m, \mathbf{x} \rangle)$ is independent of $(f_1, \ldots, f_m)$. But since $x \notin \mathbf{V}$, $(\langle \mathbf{w}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{w}_m, \mathbf{x} \rangle)$ is independent of $(\mathbf{x}, \mathbf{u}_1, \ldots, \mathbf{u}_m)$ and hence is also independent of $(f_1, \ldots, f_m)$.
2. For $i \in [m]$ define the event X_i as follows: $X_i = 1$ iff $f_i = \langle \mathbf{w}_i, \mathbf{x} \rangle$.
3. For all $i \in [m]$, $\mu := \Pr[X_i = 1] \leqslant \frac{1}{q}$. Here we make use of the fact that $x \notin \mathbf{V}$ (implying that f_i and $\langle \mathbf{w}_i, \mathbf{x} \rangle$ are independent), and the fact that $x \notin \mathbf{V}$ which implies $x \neq 0^n$, and hence$\langle \mathbf{w}_i, \mathbf{x} \rangle$ is uniformly random.

$\mathcal{A}$ outputs 0 iff $X := \frac{X_1 + \cdots + X_m}{m} \geqslant \frac{1}{q} + \frac{1}{4p}$. Let $\epsilon = \frac{1}{4p}$. Since $m = 64(\kappa + \rho)p^2$, we have $\frac{m\epsilon^2}{4} = k + \rho$. Invoking the Chernoff bound (Lemma 3.1),

$$\Pr[X \geqslant \frac{1}{q} + \frac{1}{4p}] \leqslant \Pr[|X - \mu| \geqslant \frac{1}{4p}] \leqslant \frac{2}{e^{\frac{m\epsilon^2}{4}}} = \frac{2}{e^{\kappa+\rho}}, \tag{23}$$

as desired. □

D Standard Security Notions for IPFE

We recall the standard indistinguishability-based security notions for function-hiding IPFE.

Definition D.1 (FH-IND-CPA security). Fix an oracle O and let $\mathcal{E}^O = (\mathrm{Setup}^O, \mathrm{KGen}^O, \mathrm{Enc}^O, \mathrm{Dec}^O)$ be an IPFE over $\mathbb{Z}_q^n$. The IND-CPA security game samples $\mathsf{msk} \leftarrow \mathrm{Setup}^O(1^\kappa)$ and a bit $b \leftarrow \{0,1\}$. The adversary $\mathcal{A}^O$ can make queries of the following types adaptively.

- **LoR key corruption queries**: The adversary $\mathcal{A}$ submits $(\mathbf{u}_0, \mathbf{u}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$ and receives $\mathrm{KGen}^O(\mathsf{msk}, \mathbf{u}_b)$.
- **LoR encryption queries**: The adversary $\mathcal{A}$ submits $(\mathbf{x}_0, \mathbf{x}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$ and receives $\mathrm{Enc}^O(\mathsf{msk}, \mathbf{x}_b)$.

At the end, $\mathcal{A}$ outputs a bit b' and wins if $b = b'$. We say that $\mathcal{A}$ is admissible if for any LoR key corruption query $(\mathbf{u}_0, \mathbf{u}_1)$ and for any LoR encryption query $(\mathbf{x}_0, \mathbf{x}_1)$ it ever makes, $\langle \mathbf{u}_0, \mathbf{x}_0 \rangle = \langle \mathbf{u}_1, \mathbf{x}_1 \rangle$. We say that $\mathcal{E}^O$ is FH-IND-CPA secure if any admissible PPT $\mathcal{A}$ wins the game with probability at most $1/2 + \mathrm{negl}(\kappa)$.

We say that $\mathcal{A}$ is non-function hiding admissible, if it is admissible in the above sense and only makes LoR key corruption queries with $\mathbf{u}_0 = \mathbf{u}_1$. We say that $\mathcal{E}^O$ is IND-CPA secure if any non-function hiding admissible PPT $\mathcal{A}$ wins the game with probability at most $1/2 + \mathrm{negl}(\kappa)$.

We say that the IPFE is selective (FH)-IND-CPA secure, if it satisfies the above definition with the modification that the adversary $\mathcal{A}$ has to make all of its queries in one shot (i.e. the adversary provides the inputs for all oracle queries before receiving the output of any query).

D.1 Relations Between Security Notions

Next, we show that this standard FH-IND-CPA security notion for IPFE imply (U, V)-key hiding, (U, V)-plaintext hiding (for any valid pair (U, V)), and V-plaintext hiding, and that (U, V)-key hiding implies V-key hiding.

Theorem D.2 *A selective (FH-)IND-CPA secure IPFE for $\mathbb{Z}_q^n$ is also (U, V)-plaintext hiding for any valid pair (U, V).*

Proof. We first describe the reduction. The reduction simulates straight key corruption queries by sampling $\mathbf{u} \leftarrow \mathsf{U}$ making a LoR key corruption query in the IND-CPA game on $(\mathbf{u}, \mathbf{u})$ to retrieve sk and returning $(\mathbf{u}, \mathsf{sk})$ to the adversary. For the special key corruption query, it proceeds similarly but samples the vector as $\tilde{\mathbf{u}} \leftarrow \mathbb{Z}_q^n$ conditioned on $(\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^\perp \not\subseteq \mathsf{V}$. The reduction simulates each straight encryption query by sampling $\mathbf{v} \leftarrow \mathsf{V}$, making a LoR encryption query in the IND-CPA game on $(\mathbf{v}, \mathbf{v})$ to obtain ct, and returning $(\mathbf{v}, \mathsf{ct})$. The reduction simulates each LoR encryption query by sampling $\mathbf{x}' \leftarrow (\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^\perp$ and $\mathbf{v} \leftarrow \mathsf{V}$, making a LoR encryption query in the IND-CPA game on $(\mathbf{v}, \mathbf{v} + \mathbf{x}')$ to obtain ct, and returning $(\langle \tilde{\mathbf{u}}, \mathbf{v} \rangle, \mathsf{ct})$. Finally, the reduction outputs the same bit as the adversary.

First, note that the restriction $(\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^\perp \not\subseteq \mathsf{V}$ implies $\mathsf{V} + (\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^\perp = \mathbb{Z}_q^n$. Thus, $\mathbf{v} + \mathbf{x}'$, where $\mathbf{x}' \leftarrow (\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^\perp$ and $\mathbf{v} \leftarrow \mathsf{V}$, is distributed like a uniformly random vector from $\mathbb{Z}_q^n$. This shows that the reduction simulates the plaintext hiding game with the same challenge bit that is used in the FH-IND-CPA game.

Moreover, the reduction is an admissible FH-IND-CPA adversary because for any vector $\mathbf{u} \in \mathsf{U} \cup \{\mathbf{y}$ and for each vector $\mathbf{x}' \in (\mathsf{Span}(\mathsf{U} \cup \{\mathbf{y}\}))^{\perp}, \mathbf{v} \in \mathsf{V}$ we have

$$\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{u}, \mathbf{v} + \mathbf{x}' \rangle.$$

This also shows that the reduction correctly simulates the inner product in the LoR encryption queries by returning $\langle \tilde{\mathbf{u}}, \mathbf{v} \rangle$ (instead of $\langle \tilde{\mathbf{u}}, \mathbf{v} + b\mathbf{x}' \rangle$ for the unknown challenge bit b).

Since the inputs for all queries the reduction makes a derived from vectors the reduction samples itself from fixed subsets of $\mathbb{Z}_q^n$, the reduction can sample all these vectors in the beginning and make all queries in one shot. □

Theorem D.3 *A selective FH-IND-CPA secure IPFE for $\mathbb{Z}_q^n$ is also (U, V)-key hiding.*

Proof. The reduction is straightforward. The reduction simulates straight key corruption queries by sampling $\mathbf{u} \leftarrow \mathsf{U}$ making a LoR key corruption query on $(\mathbf{u}, \mathbf{u})$ in the FH-IND-CPA game to retrieve sk and returning $(\mathbf{u}, \mathsf{sk})$ to the adversary. It simulates left-or-right key corruption queries by sampling $\mathbf{w} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{v} \leftarrow \mathsf{V}^{\perp}$, making a LoR key corruption query on $(\mathbf{w}, \mathbf{w} + \mathbf{v})$ in the FH-IND-CPA game to retrieve sk and returning $(\mathbf{w}, \mathbf{w} + \mathbf{v}, \mathsf{sk})$ to the adversary. Encryption queries are simulated by sampling $\mathbf{x} \leftarrow \mathsf{V}$, making a $(\mathbf{x}, \mathbf{x})$ LoR encryption query in the FH-IND-CPA game to retrieve ct, and sending $(\mathbf{x}, \mathsf{ct})$ to the adversary. Finally, the reduction outputs the same bit as the adversary.

It is clear that the reduction perfectly simulates the key hiding game perfectly with the same challenge bit as in the FH-IND-CPA game. The reduction is an admissible FH-IND-CPA adversary because for any pair of vectors $(\mathbf{w}, \mathbf{w} + \mathbf{v})$ the reduction submits to a LoR key corruption oracle and any pair of vectors $(\mathbf{x}, \mathbf{x})$ the reduction submits to a LoR encryption oracle we have

$$\langle \mathbf{w}, \mathbf{x} \rangle = \langle \mathbf{w} + \mathbf{v}, \mathbf{x} \rangle,$$

since $\mathbf{v} \in \mathsf{V}^{\perp}$ and $\mathbf{x} \in \mathsf{V}$.

Since the inputs for all queries the reduction makes a derived from vectors the reduction samples itself from fixed subsets of $\mathbb{Z}_q^n$, the reduction can sample all these vectors in the beginning and make all queries in one shot. □

Next, we show that (U, V)-key hiding and (U, V)-plaintext hiding imply V-key hiding and V-plaintext hiding, respectively.

Theorem D.4 *A selective (FH-)IND-CPA secure IPFE for $\mathbb{Z}_q^n$ is also V-plaintext hiding for any $\mathsf{V} \subseteq \mathbb{Z}_q^n$.*

Proof. We first describe the reduction. The reduction simulates the key corruption query for $\mathbf{y}$ with a LoR key corruption query on $(\mathbf{y}, \mathbf{y})$ and each straight encryption query by sampling $\mathbf{x} \leftarrow \mathbb{Z}_q$, making a LoR encryption query in the FH-IND-CPA game on $(\mathbf{x}, \mathbf{x})$ to obtain ct, and returning $(\mathbf{x}, \mathsf{ct})$. The reduction simulates each LoR encryption query on $\mathbf{x}$ in the plaintext hiding game by sampling $\mathbf{x}' \leftarrow \mathbf{y}^{\perp}$ and $\mathbf{v} \leftarrow \mathsf{V}$, making a LoR encryption query in the FH-IND-CPA game on $(\mathbf{v}, \mathbf{v} + \mathbf{x}')$ to obtain ct, and returning $(\langle \mathbf{y}, \mathbf{v} \rangle, \mathsf{ct})$. Finally, the reduction outputs the same bit as the adversary.

First, note that $\mathbf{y}^{\perp} \neq \mathsf{V}$. Otherwise we would have $\mathsf{V} = \{\mathbf{w} \in \mathbb{Z}_q^n \mid \langle \mathbf{w}, \mathbf{y} \rangle = 0\}$ and thus in particular $\langle \mathbf{v}, \mathbf{y} \rangle = 0$ for all $\mathbf{v} \in \mathsf{V}$. This implies $\mathbf{y} \in \mathsf{V}^{\perp}$, which is not allowed in the plaintext hiding game.

Since $\mathbf{y}^\perp$ and V are two different $(n-1)$-dimensional subspaces of $\mathbb{Z}_q^n$, we have $\mathbf{y}^\perp + \mathsf{V} = \mathbb{Z}_q^n$ and thus for $\mathbf{x}' \leftarrow \mathbf{y}^\perp$ and $\mathbf{v} \leftarrow \mathsf{V}$ the vector $\mathbf{v} + \mathbf{x}'$ is uniformly random in $\mathbb{Z}_q^n$ This shows that the reduction simulates the plaintext hiding game with the same challenge bit that is used in the FH-IND-CPA game. Moreover, the reduction is an admissible FH-IND-CPA adversary because for the vector $\mathbf{u} \notin \mathsf{V}^\perp$ from $\mathcal{A}$'s key corruption query and for each vector $\mathbf{x}' \in \mathbf{y}^\perp, \mathbf{v} \in \mathsf{V}$ sampled in an LoR encryption query of $\mathcal{A}$ we have

$$\langle \mathbf{y}, \mathbf{v} \rangle = \langle \mathbf{y}, \mathbf{v} + \mathbf{x}' \rangle.$$

This also shows that the reduction correctly simulates the inner product in the LoR encryption queries by always returning $\langle \mathbf{y}, \mathbf{v} \rangle$. □

Theorem D.5 *A (U, V)-key hiding IPFE is also V-key hiding.*

Proof. The reduction is trivial. The V-key hiding gives the adversary two oracles (for LoR key corruption and straight encryption), and both of these oracles are also present in the (U, V)-key hiding game. □

E Bounded-Collusion FH-IPFE

E.1 Technical overview

Our impossibility result only rules out function-hiding IPFE in the unbounded-collusion setting, because the number of key queries the information-theoretic adversary makes in the end has to grow with the number of bits in the master secret key. Similarly, we need V-plaintext hiding for an unbounded number of ciphertexts to ensure that the compilers described above works.

Indeed, if we bound the number of functional keys (or, equivalently, the number of ciphertexts[16]) an adversary can obtain in the security experiment (one-side bounded), we can achieve function-hiding IPFE from generic groups.

One way to achieve this is to use the generic transformation of [20], which turns an FE schemes for bounded-size circuits into function-hiding FE scheme for bounded-size circuits. When we combine this result with [13], which constructs FE for bounded-size circuits in the bounded-collusion setting assuming just one-way functions, we get function-hiding FE for bounded-size circuits assuming just one-way functions.[17]

However, if the goal is just to construct function-hiding *inner-product* FE, this approach seems overly complicated and very inefficient, since it uses a Boolean circuit to compute the inner product. Thus, we present a much simpler transformation that turns any one-side bounded-collusion IPFE into a one-side bounded-collusion FH-IPFE (see Section E.3).

We present our transformation here first for the simpler 1-key setting, where the adversary is only allowed to make a single functional key query. Later, we explain how we translate the one-side bounded-collusion setting, where any polynomial bound on the number of keys can be used.

[16] Thanks to the symmetry of the inner product, we can swap the roles of keys and ciphertexts for function-hiding IPFE. This allows us to trade a bound for the functional keys for a bound on the ciphertexts and vice versa.

[17] Technically, this does not imply bound-collusion FH-IPFE in Maurer's GGM, because in this model the group cannot be used to obtain a one-way function from bitstrings to bitstrings. However, with the transformation we propose, we can get FH-IPFE in Maurer's GGM.

In our 1-key FH-IPFE, we store in the master secret key an additional vector $\hat{\mathbf{y}}$. A functional key for a vector $\mathbf{y}$ is a functional secret key of the underlying IPFE for the vector $\begin{pmatrix} \mathbf{y}+\hat{\mathbf{y}} \\ -1 \end{pmatrix}$. A ciphertext for vector $\mathbf{x}$ is then generated by encrypting[18] $\begin{pmatrix} \mathbf{x} \\ \langle \mathbf{x}, \hat{\mathbf{y}} \rangle \end{pmatrix}$. We make no changes to decryption.

Informally speaking, the above scheme is function-hiding because instead we encrypt the vector $\mathbf{y}$ with a one-time pad, before we generate the secret keys for it. A ciphertext reveals with the component $\langle \mathbf{x}, \hat{\mathbf{y}} \rangle$ partial information about $\hat{\mathbf{y}}$ and thus indirectly also about $\mathbf{y}$. However, it is not difficult to see that is equivalent to revealing $\langle \mathbf{x}, \mathbf{y} \rangle$, which is allowed to be leaked about $\mathbf{y}$.

To extend this to bounded-collusion IPFE, we essentially replace the one-time pad encryption of $\mathbf{y}$ with a "many-time pad", where $\mathbf{y}$ is blinded not by a fixed vector $\hat{\mathbf{y}}$ but by a random linear combination $\sum_{i=1}^{\ell} r_i \hat{\mathbf{y}}_i$ of many vectors $\hat{\mathbf{y}}_i$ that are stored in the master secret key. If the parameter ℓ is large enough, this hides $\mathbf{y}$ statistically even when the coefficients $r_i \in \mathbb{Z}_q$ are released.

Finally, we observe that we can build with similar techniques an information-theoretic (non-function-hiding) IPFE where the number of ciphertexts the adversary obtains is bounded (see Section E.4). If we apply our compiler to that scheme, we get an information-theoretic two-side bounded collusion (where both ciphertexts and functional keys are bounded) FH-IPFE.

E.2 Definitions

We define two variants of bounded-collusion FH-IPFE:

- Bounded collusion FH-IPFE, where the number of functional keys[19] the adversary can query is a priori bounded.
- Fully bounded collusion FH-IPFE, where both the number of keys and ciphertexts the adversary can query is bounded.

Definition E.1 (Bounded collusion IPFE). A *bounded collusion* private-key IPFE scheme $\mathcal{E} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ for the module $\mathbb{Z}_q^n$ is defined identically to regular IPFE (see Definition 3.2), except that Setup gets as additional parameter 1^{Q_k}.

Definition E.2 (Fully bounded collusion IPFE). A *two-side bounded collusion* private-key IPFE scheme $\mathcal{E} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ for the module $\mathbb{Z}_q^n$ is defined identically to regular IPFE (see Definition 3.2), except that Setup gets as additional parameters $1^{Q_k}, 1^{Q_e}$.

In the following, the parts in parentheses apply only to the two-side bounded-collusion case.

Correctness is defined for (two-side) bounded collusion IPFE identically to regular IPFE. The security notions IND-CPA and FH-IND-CPA (Definition D.1) are defined identically to regular IPFE except for the following changes:

1. The adversary sends 1^{Q_k} (and 1^{Q_e}) to the game. The master secret key is generated with respect to these parameters.
2. The adversary can make at most Q_k (LoR) key corruption queries and, for two-side bounded collusion IPFE, at most Q_e (LoR) encryption queries.

[18] It is important that also $\langle \mathbf{x}, \hat{\mathbf{y}} \rangle$ gets encrypted, and not just send along with the ciphertext, because $\langle \mathbf{x}, \hat{\mathbf{y}} \rangle$ can reveal secret information about $\mathbf{x}$. For example, it can be used to check if $\mathbf{x} = 0$ or not with high probability.

[19] For FH-IPFE, the notion where we bound the number of encryptions the adversary can query instead, is equivalent.

E.3 Bounded-Collusion IPFE to Bounded-Collusion FH-IPFE

Our first result is a generic transformation that turns a (non-function hiding) one-side bounded-collusion IPFE into a one-side bounded-collusion FH-IPFE.

Let $\mathcal{E} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ be an IND-CPA secure one-side (two-side) bounded-collusion IPFE. We describe how to build from this an FH-IND-CPA secure (two-side) bounded-collusion IPFE $\mathcal{E}' = (\mathrm{Setup}', \mathrm{KGen}', \mathrm{Enc}', \mathrm{Dec}')$. The transformation uses a parameter $\ell = \ell(\kappa)$. It achieves security, for example, for $\ell := Q_k + \kappa$ for any modulus q or for $\ell := Q_k$ for any super-polynomial prime q. The transformation preserves the output encoding. When $\mathcal{E}'$ is an IPFE for $\mathbb{Z}_q^n$, we need $\mathcal{E}$ to be an IPFE for $\mathbb{Z}_q^{n+\ell}$.

- $\mathrm{Setup}'(1^\kappa)$ runs $\mathrm{Setup}(1^\kappa)$ to obtain a master secret key msk for $\mathcal{E}$ and samples additionally ℓ vectors $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell \leftarrow \mathbb{Z}_q^n$. It returns $\mathsf{msk}' := (\mathsf{msk}, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell)$
- $\mathrm{KGen}'(\mathsf{msk}' = (\mathsf{msk}, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell), \mathbf{y})$ samples $r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_q$ and returns

$$\mathrm{KGen}(\mathsf{msk}, \begin{pmatrix} \mathbf{y} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{y}}_i \\ -r_1 \\ \vdots \\ -r_\ell \end{pmatrix}).$$

- $\mathrm{Enc}'(\mathsf{msk}' = (\mathsf{msk}, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell), \mathbf{x})$ returns

$$\mathrm{Enc}(\mathsf{msk}, \begin{pmatrix} \mathbf{x} \\ \langle \hat{\mathbf{y}}_1, \mathbf{x} \rangle \\ \vdots \\ \langle \hat{\mathbf{y}}_\ell, \mathbf{x} \rangle \end{pmatrix}).$$

- $\mathrm{Dec}'(\mathsf{sk}, \mathsf{ct}) = \mathrm{Dec}(\mathsf{sk}, \mathsf{ct})$.

Theorem E.3 *If the IPFE $\mathcal{E}$ is ν-correct, then the IPFE $\mathcal{E}'$ is also $\nu - 2/q^n$-correct.*

Proof. For uniformly random $\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell \leftarrow \mathbb{Z}_q^n, r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_q$ the vectors $\mathbf{x}', \mathbf{y}'$ are $2/q^n$ close to uniformly random and satisfy

$$\Big\langle \underbrace{\begin{pmatrix} \mathbf{y} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{y}}_i \\ -r_1 \\ \vdots \\ -r_\ell \end{pmatrix}}_{=:\mathbf{y}'}, \underbrace{\begin{pmatrix} \mathbf{x} \\ \langle \hat{\mathbf{y}}_1, \mathbf{x} \rangle \\ \vdots \\ \langle \hat{\mathbf{y}}_\ell, \mathbf{x} \rangle \end{pmatrix}}_{=:\mathbf{x}'} \Big\rangle = \langle \mathbf{y} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{y}}_i, \mathbf{x} \rangle - \sum_{i=1}^{\ell} r_i \langle \hat{\mathbf{y}}_i, \mathbf{x} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle. \qquad \square$$

We call parameters q, Q_k, ℓ valid if $\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{Q_k \times \ell}}[\mathsf{Span}(\mathbf{A}) = \mathbb{Z}_q^{Q_k}] \geqslant 1 - \mathrm{negl}(\kappa)$ for a negligible function negl. Here $\mathsf{Span}(\mathbf{A})$ denotes the $\mathbb{Z}_q$ module spanned by the columns of $\mathbf{A}$. We show later how to find for all q, Q_k a parameter ℓ that is just slightly larger than Q_k such that q, Q_k, ℓ is valid.

The following lemma is the main tool to argue security of our transformation (and the fully bound collusion construction in the next section).

Lemma E.4 *Fix a submodule $\mathsf{V} \subseteq \mathbb{Z}_q^n$. For uniformly random $r_{1,1}, \ldots, r_{Q_k,\ell} \leftarrow \mathbb{Z}_q, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell \leftarrow \mathsf{V}$, the vectors*

$$\begin{pmatrix} \sum_{i=1}^{\ell} r_{1,i}\hat{\mathbf{y}}_i \\ -r_{1,1} \\ \vdots \\ -r_{1,\ell} \end{pmatrix}, \ldots, \begin{pmatrix} \sum_{i=1}^{\ell} r_{Q_k,i}\hat{\mathbf{y}}_i \\ -r_{Q_k,1} \\ \vdots \\ -r_{Q_k,\ell} \end{pmatrix}$$

are statistically close to uniformly random from the space $\mathsf{V} \times \mathbb{Z}_q^\ell$

Proof. Note that if the matrix

$$\mathbf{R} := \begin{pmatrix} r_{1,1} & \cdots & r_{1,\ell} \\ \vdots & \ddots & \vdots \\ r_{Q_k,1} & \cdots & r_{Q_k,\ell} \end{pmatrix} \in \mathbb{Z}_q^{Q_k \times \ell}$$

spans $\mathbb{Z}_q^{Q_k}$ with overwhelming probability by the assumption that the q, Q_k, ℓ are valid. For the following we assume $\mathsf{Span}(\mathbf{R}) = \mathbb{Z}_q^{Q_k}$.

Consider then the function

$$\begin{aligned} f : \mathbb{Z}_q^\ell &\to \mathbb{Z}_q^{Q_k} \\ \mathbf{z} &\mapsto \mathbf{R}\mathbf{z}. \end{aligned}$$

Each vector in $\mathbb{Z}_q^{Q_k}$ has a preimage of the same size (namely, $q^{\ell - Q_k}$) under this function, because $\mathsf{Span}(\mathbf{R}) = \mathbb{Z}_q^{Q_k}$. In particular, for fixed $\mathbf{R}$ and uniformly random $\mathbf{z}$, the vector $\mathbf{R}\mathbf{z}$ is uniformly random.

Now fix $\mathbf{b}_1, \ldots, \mathbf{b}_{\tilde{n}} \in \mathbb{Z}_q^n$ with $\mathsf{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_{\tilde{n}}) = \mathsf{V}$. For each $i \in [\ell]$ the vector $\hat{\mathbf{y}}_i$ is distributed as $\sum_{j=1}^{\tilde{n}} \alpha_{i,j}\mathbf{b}_j$ for $\alpha_{i,j} \leftarrow \mathbb{Z}_q^n$. In matrix notation, this becomes

$$(\hat{\mathbf{y}}_1 \mid \cdots \mid \hat{\mathbf{y}}_\ell) = (\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_{\tilde{n}}) \cdot \mathbf{A}^\top \quad \text{where} \quad \mathbf{A} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,\tilde{n}} \\ \vdots & \ddots & \vdots \\ \alpha_{\ell,1} & \cdots & \alpha_{\ell,\tilde{n}} \end{pmatrix}$$

For fixed $\mathbf{R} \in \mathbb{Z}_q^{Q_k \times \ell}$ with $\mathsf{Span}(\mathbf{R}) = \mathbb{Z}_q^{Q_k}$ and uniformly random $\mathbf{A} \leftarrow \mathbb{Z}_q^{\ell \times \tilde{n}}$, the matrix $\mathbf{R}\mathbf{A}$ is uniformly random in $\mathbb{Z}_q^{Q_k \times \tilde{n}}$ and thus

$$\begin{aligned} \left(\sum_{i=1}^{\ell} r_{1,i}\hat{\mathbf{y}}_i \;\middle|\; \cdots \;\middle|\; \sum_{i=1}^{\ell} r_{Q_k,i}\hat{\mathbf{y}}_i\right) &= \mathbf{R}\begin{pmatrix} \alpha_{1,1} \cdots \alpha_{1,\tilde{n}} \\ \vdots \\ \alpha_{1,1} \cdots \alpha_{1,\tilde{n}} \end{pmatrix} = (\hat{\mathbf{y}}_1 \mid \cdots \mid \hat{\mathbf{y}}_\ell) \cdot \mathbf{R}^\top \\ &= (\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_{\tilde{n}})\mathbf{A}^\top\mathbf{R}^\top \qquad \text{where} \quad \mathbf{R} = \begin{pmatrix} r_{1,1} & \cdots & r_{1,\ell} \\ \vdots & \ddots & \vdots \\ r_{Q_k,1} & \cdots & r_{Q_k,\ell} \end{pmatrix} \end{aligned}$$

are uniformly random vectors from V. □

We show that the transformation achieves weak FH-IND-CPA (wFH-IND-CPA) security, which is defined like FH-IND-CPA security (Definition D.1), except that and adversary is admissible if for any LoR key corruption query $(\mathbf{u}_0, \mathbf{u}_1)$ and for any LoR encryption query $(\mathbf{x}_0, \mathbf{x}_1)$ it ever makes, $\langle \mathbf{u}_0, \mathbf{x}_0 \rangle = \langle \mathbf{u}_0, \mathbf{x}_1 \rangle = \langle \mathbf{u}_1, \mathbf{x}_1 \rangle$ (instead of requiring only $\langle \mathbf{u}_0, \mathbf{x}_0 \rangle = \langle \mathbf{u}_1, \mathbf{x}_1 \rangle$). This notion can be generically lifted to full FH-IND-CPA security [31, Section 5.7].

Theorem E.5 *If $\mathcal{E}$ is (two-side) bounded-collusion IND-CPA secure and n, q, ℓ is valid, then $\mathcal{E}'$ is (two-side) bounded-collusion wFH-IND-CPA secure.*

Proof. Assume there exists a PPT adversary $\mathcal{A}$ against the wFH-IND-CPA security of $\mathcal{E}'$. We show how to turn this into a PPT adversary $\mathcal{B}$ against the IND-CPA security of $\mathcal{E}$.

The adversary $\mathcal{B}$ runs $\mathcal{A}$ internally and simulates the transformation. In more detail, the adversary $\mathcal{A}$ initially samples $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell \leftarrow \mathbb{Z}_q^n$ and retrieves Q_k (and Q_e) from $\mathcal{A}$. Whenever $\mathcal{A}$ makes a key corruption query for the vectors $(\mathbf{y}_0, \mathbf{y}_1)$, the adversary $\mathcal{B}$ samples $r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_q$, makes a key corruption query for

$$\begin{pmatrix} \mathbf{y}_0 + \sum_{i=1}^{\ell} r_i \hat{\mathbf{y}}_i \\ -r_1 \\ \vdots \\ -r_\ell \end{pmatrix},$$

and forwards the result to $\mathcal{A}$. Whenever $\mathcal{A}$ makes an encryption query for the vector $(\mathbf{x}_0, \mathbf{x}_1)$, the adversary $\mathcal{B}$ makes an encryption query for

$$\left(\begin{pmatrix} \mathbf{x}_0 \\ \langle \hat{\mathbf{y}}_1, \mathbf{x}_0 \rangle \\ \vdots \\ \langle \hat{\mathbf{y}}_\ell, \mathbf{x}_0 \rangle \end{pmatrix}, \begin{pmatrix} \mathbf{x}_1 \\ \langle \hat{\mathbf{y}}_1, \mathbf{x}_1 \rangle \\ \vdots \\ \langle \hat{\mathbf{y}}_\ell, \mathbf{x}_1 \rangle \end{pmatrix} \right)$$

and forwards the result to $\mathcal{A}$. In the end, $\mathcal{B}$ outputs the same bit as $\mathcal{A}$.

When we play the IND-CPA game for $\mathcal{E}$ with $\mathcal{B}$, it is clear that $\mathcal{B}$ simulates the encryption queries with the same challenge bit for $\mathcal{A}$ as they would be answered in the FH-IND-CPA game for $\mathcal{E}'$. It is also clear that if $b = 0$ $\mathcal{B}$ simulates the key corruption queries for $\mathcal{A}$ correctly. Let $(\mathbf{y}_{1,0}, \mathbf{y}_{1,1}), \ldots, (\mathbf{y}_{Q_k,0}, \mathbf{y}_{Q_k,1})$ be the inputs of all encryption queries that $\mathcal{A}$ makes. If $b = 1$, we argue that from $\mathcal{A}$'s point of view for all $j \in [Q_k]$

$$\begin{pmatrix} \mathbf{y}_{j,0} + \sum_{i=1}^{\ell} r_{j,i} \hat{\mathbf{y}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{y}_{j,1} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{y}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix}$$

are simultaneously statistically close and thus $\mathcal{B}$ simulates these key queries correctly, too, with overwhelming probability.

Therefore, let $(\mathbf{x}_{1,0}, \mathbf{x}_{1,1}), \ldots, (\mathbf{x}_{q,0}, \mathbf{x}_{q,1})$ be the inputs of all encryption queries that $\mathcal{A}$ makes and $\mathsf{V} := \{\mathbf{v} \in \mathbb{Z}_q^n \mid \forall i \in [q] : \langle \mathbf{v}, \mathbf{x}_{i,1} \rangle = 0\}$. Each $\hat{\mathbf{y}}_i$ $(i \in [\ell])$ is distributed as $\hat{\mathbf{y}}_i = \bar{\mathbf{y}}_i + \tilde{\mathbf{y}}_i$ where $\bar{\mathbf{y}}_i \leftarrow \mathsf{V}$ and $\tilde{\mathbf{y}}_i \leftarrow \mathbb{Z}_q^n$. The encryption queries of $\mathcal{A}$ are statistically independent of $\bar{\mathbf{y}}_i$. Thus, in $\mathcal{A}$'s

view for all $j \in [Q_k]$ simultaneously

$$\begin{pmatrix} \mathbf{y}_{j,0} + \sum_{i=1}^{\ell} r_{j,i}\bar{\mathbf{y}}_i + \tilde{\mathbf{y}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix} \overset{(1)}{\approx_s} \begin{pmatrix} \mathbf{y}_{j,0} + \sum_{i=1}^{\ell} r_i\tilde{\mathbf{y}}_i \\ -r_1 \\ \vdots \\ -r_\ell \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\overset{(2)}{\equiv} \begin{pmatrix} \mathbf{y}_{j,1} + \sum_{i=1}^{\ell} r_i\tilde{\mathbf{y}}_i \\ -r_1 \\ \vdots \\ -r_\ell \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \overset{(3)}{\approx_s} \begin{pmatrix} \mathbf{y}_{j,1} + \sum_{i=1}^{\ell} r_{j,i}\bar{\mathbf{y}}_i + \tilde{\mathbf{y}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix}$$

where $\mathbf{v} \leftarrow \mathsf{V}$. Here step (1) and (3) follow from Lemma E.4 and (2) follows from $\mathbf{y}_{j,1} - \mathbf{y}_{j,0} \in \mathsf{V}$ (which in turn follows from the admissibility of a weak FH-IND-CPA adversary).

This shows that the game that $\mathcal{B}$ simulates for $\mathcal{A}$ (when $\mathcal{B}$ plays the IND-CPA game for $\mathcal{E}$) is statistically close to the wFH-IND-CPA game for $\mathcal{E}$ with the same challenge bit. □

The following lemma is helpful to determine which parameters q, Q_k, ℓ are valid.

Lemma E.6 *For $q, Q_k, \ell \in \mathbb{N}_+$, $n \leqslant \ell$ and $q \geqslant 2$, where $q = p_1^{e_1} \cdot \dots \cdot p_k^{e_k}$ is the prime factorization of q,*

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{Q_k \times \ell}}[\mathsf{Span}(\mathbf{A}) = \mathbb{Z}_q^{Q_k}] \geqslant 1 - \frac{p_1^{\ell - Q_k}}{p_1 - 1} \cdot \dots \cdot \frac{p_k^{\ell - Q_k}}{p_k - 1}$$

Proof. We proof the statement first for the case that q is a prime.

The probability that a uniformly random ℓ-dimensional vector is inside a $k \leqslant \ell$ dimensional subspace is $q^{k-\ell}$. With the same probability the k-th row vector of $\mathbf{A}$ is not in the span of the first $k-1$ row vectors of $\mathbf{A}$. Applying this to all n row vectors yields

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times \ell}}[\mathsf{Span}(\mathbf{A}) \neq \mathbb{Z}_q^{Q_k}] = q^{-\ell} + \dots + q^{n-1-\ell} \leqslant \frac{q^{n-\ell}}{q-1}$$

Next, we consider the case where $q = p^e$ is a prime power. We have $\mathsf{Span}(\mathbf{A}) = \mathbb{Z}_q^{Q_k}$ iff $\mathsf{Span}(\mathbf{A} \bmod p) = \mathbb{Z}_p^{Q_k}$. Thus, exploiting that for $a \leftarrow \mathbb{Z}_q$ that $a \bmod p$ is distributed uniformly at random in $\mathbb{Z}_p$ and the bound for prime moduli, we get $\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times \ell}}[\mathsf{Span}(\mathbf{A}) \neq \mathbb{Z}_q^{Q_k}] \leqslant \frac{p^{n-\ell}}{p-1}$.

The case for general q then follows directly by applying the Chinese Remainder Theorem. □

When q is a super-polynomially large prime, $1/(q-1)$ is neglible and thus we can choose $\ell = Q_k$. For general moduli, we can always use the following bound

$$\frac{p_1^{\ell - Q_k}}{p_1 - 1} \cdot \dots \cdot \frac{p_k^{\ell - Q_k}}{p_k - 1} \leqslant \frac{p_1^{\ell - Q_k}}{2p_1} \cdot \dots \cdot \frac{p_k^{\ell - Q_k}}{2p_k} = \frac{p_1^{\ell - Q_k - 1} \cdot \dots \cdot p_k^{\ell - Q_k - 1}}{2} \leqslant 2^{\ell - Q_k - 2}$$

to see that $\ell = Q_k + \kappa$ is a valid parameter for any Q_k and q.

Other security notions. With essentially the same argument we can show that the transformation can also achieve other function-hiding security notions from their non-function hiding counterparts, such as (weak) selective FH-IND-CPA security from selective IND-CPA and function-hiding simulation security form regular simulation security for several variants of simulation security (selective/adaptive, single/multi-challenge).

E.4 Information-Theoretic Fully Bounded-Collusion FH-IPFE

Here we describe how to build a two-side bounded-collusion IPFE that achieves SIM-CPA security information-theoretically. With the transformation from Section E.3 we can then get a two-side bounded-collusion IPFE with information-theoretic FH-SIM-CPA security. As before, we describe the scheme with a parameter ℓ that can be selected as in the transformation. The IPFE scheme works as follows:

- $\mathrm{Setup}(1^\kappa)$ samples ℓ vectors $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_\ell \leftarrow \mathbb{Z}_q^n$ and returns $\mathsf{msk} := (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_\ell)$.
- $\mathrm{KGen}(\mathsf{msk} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_\ell), \mathbf{y})$ returns $(\mathbf{y}, \langle \mathbf{y}, \hat{\mathbf{x}}_1 \rangle, \ldots, \langle \mathbf{y}, \hat{\mathbf{x}}_\ell \rangle)$
- $\mathrm{Enc}(\mathsf{msk} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_\ell), \mathbf{x})$ samples $r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_q$ and returns $(\mathbf{x} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{x}}_i, -r_1, \ldots, -r_\ell)$
- $\mathrm{Dec}'(\mathsf{sk} = (\mathbf{y}, \langle \mathbf{y}, \hat{\mathbf{x}}_1 \rangle, \ldots, \langle \mathbf{y}, \hat{\mathbf{x}}_\ell \rangle), \mathsf{ct} = (\mathbf{x} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{x}}_i, r_1, \ldots, r_\ell))$ returns $\langle \mathbf{y}, \mathbf{x} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{x}}_i \rangle - \sum_{i=1}^{\ell} r_i \langle \mathbf{y}, \hat{\mathbf{x}}_i \rangle$.

Theorem E.7 *The two-side bounded collusion IPFE described above is* 1*-correct.*

Proof. The linearity of the inner product yields

$$\langle \mathbf{y}, \mathbf{x} + \sum_{i=1}^{\ell} r_i \hat{\mathbf{x}}_i \rangle - \sum_{i=1}^{\ell} r_i \langle \mathbf{y}, \hat{\mathbf{x}}_i \rangle = \langle \mathbf{y}, \mathbf{x} \rangle.$$

□

The IPFE from above is secure as long as the number of encryption queries is bounded. The number of key corruption does not have to be bounded to achieve IND-CPA security. It only has to be bounded when we also apply the transformation from Section E.3 to achieve FH-IND-CPA security.

Theorem E.8 *The two-side bounded collusion IPFE described above is IND-CPA secure (even against unbounded adversaries) if n, q, ℓ are valid.*

Proof. Assume there exists an adversary $\mathcal{A}$ against the IND-CPA security of the IPFE scheme. Let $\mathbf{y}_1, \ldots, \mathbf{y}_{Q_k}$ be the secret key vectors that the adversary corrupts during the game, $\mathsf{V} = \{\mathbf{x} \mid \forall j \in [Q_k] : \langle \mathbf{x}, \mathbf{y}_j \rangle = 0\}$, and let $(\mathbf{x}_{1,0}, \mathbf{x}_{1,1}), \ldots, (\mathbf{x}_{q,0}, \mathbf{x}_{q,1})$ be the inputs of all encryption queries that $\mathcal{A}$ makes. Each $\hat{\mathbf{x}}_i$ $(i \in [\ell])$ is distributed as $\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \tilde{\mathbf{x}}_i$ where $\bar{\mathbf{x}}_i \leftarrow \mathsf{V}$ and $\tilde{\mathbf{x}}_i \leftarrow \mathbb{Z}_q^n$. Then, in the adversary's view, for all $j \in [Q_e]$ simultaneously,

$$\begin{pmatrix} \mathbf{x}_{j,0} + \sum_{i=1}^{\ell} r_{j,i} \bar{\mathbf{x}}_i + \sum_{i=1}^{\ell} r_{j,i} \tilde{\mathbf{x}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix} \overset{(1)}{\approx_s} \begin{pmatrix} \mathbf{x}_{j,0} + \sum_{i=1}^{\ell} r_{j,i} \tilde{\mathbf{x}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\overset{(2)}{\equiv} \begin{pmatrix} \mathbf{x}_{j,1} + \sum_{i=1}^{\ell} r_{j,i} \tilde{\mathbf{x}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix} + \begin{pmatrix} \mathbf{v} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \overset{(3)}{\approx_s} \begin{pmatrix} \mathbf{x}_{j,1} + \sum_{i=1}^{\ell} r_{j,i} \bar{\mathbf{x}}_i + \sum_{i=1}^{\ell} r_{j,i} \tilde{\mathbf{x}}_i \\ -r_{j,1} \\ \vdots \\ -r_{j,\ell} \end{pmatrix},$$

where $\mathbf{v} \leftarrow \mathsf{V}$. Here step (1) and (3) follow from Lemma E.4 and (2) follows from $\mathbf{x}_{j,1} - \mathbf{x}_{j,0} \in \mathsf{V}$, which is guaranteed by the admissibility of $\mathcal{A}$.

This shows that for any adversary $\mathcal{A}$ the distribution in the IND-CPA game is statistically close to independent of the challenge bit b, and thus it can win the IND-CPA game only with negligible advantage. □

Simulation security. With essentially the same argument we can also show that our construction achieves adaptive simulation security, and thus, with the transformation above, we can also get adaptive function-hiding simulation secure IPFE in the fully-bounded collusion setting information-theoretically.

F Combinatorial Lemmata

Lemma F.1 ([27, Lemma 4.1]). *Fix $n = n(\kappa)$ and suppose $q^{-n} \in \mathrm{negl}(\kappa)$ and $n \geqslant 3$. Let $\ell = \mathsf{poly}(\kappa)$, q be a prime number and $F : \mathbb{Z}_q^n \to 2^{[\ell]}$. Fix a constant c. Then, there exists a polynomial $t = t(\kappa)$ such that with probability at least $1 - \kappa^{-c}$*

$$F(\mathbf{y}^*) \subseteq \bigcup_{i=1}^{t} F(\mathbf{y}_i),$$

where $\mathbf{y}^* \leftarrow \mathbb{Z}_q^n$, and we sample a random $(n-1)$-dimensional subspace V subject to $\mathbf{y}^* \notin \mathsf{V}$ and we sample $\mathbf{y}_1, \ldots, \mathbf{y}_t$ all uniformly at random from V.

We adapt this lemma to also work for $n-2$ dimensional subspaces.

Lemma F.2 *Fix $n = n(\kappa)$ and suppose $q^{-n} \in \mathrm{negl}(\kappa)$ and $n \geqslant 4$. Let $\ell = \mathsf{poly}(\kappa)$, q be a prime number and $F : \mathbb{Z}_q^n \to 2^{[\ell]}$. Fix a constant c. Then, there exists a polynomial $t = t(\kappa)$ such that with probability at least $1 - \kappa^{-c}$*

$$F(\mathbf{y}^*) \subseteq \bigcup_{i=1}^{t} F(\mathbf{y}_i),$$

where $\mathbf{y}^* \leftarrow \mathbb{Z}_q^n$, and we sample a random $(n-2)$-dimensional subspace U subject to $\mathbf{y}^* \notin \mathsf{U}$ and we sample $\mathbf{y}_1, \ldots, \mathbf{y}_t$ all uniformly at random from U.

Proof. Sample $\mathbf{y}^*$ and U as in the lemma and additionally a random $(n-1)$-dimensional subspace V with $\mathsf{U} \subseteq \mathsf{V} \subseteq \mathbb{Z}_q^n$. Then there exist a polynomial t' such that

$$\Pr\left[F(\mathbf{y}^*) \subseteq \bigcup_{i=1}^{t'} F(\mathbf{y}_i) \middle| \mathbf{y}^* \in \mathsf{V} \right] \geqslant 1 - \kappa^c$$

by invoking Lemma F.1 with dimension $n-1$.

To argue the case $\mathbf{y}^* \notin \mathsf{V}$, Lemma F.1 with dimension n guarantees that for every c' there exists a polynomial t' such that for $\mathbf{z}, \ldots, \mathbf{z}_{t'} \leftarrow \mathsf{V}$ with

$$\Pr\left[F(\mathbf{y}^*) \subseteq \bigcup_{i=1}^{t'} F(\mathbf{z}_i) \middle| \mathbf{y}^* \in \mathsf{V} \right] \geqslant 1 - \kappa^{c'}.$$

Next, fix $i \in [t']$. We invoke the combinatorial lemma for dimension $n-2$ times and get that for every c'' there exists a polynomial t'' such that

$$\Pr\left[F(\mathbf{z}_i) \subseteq \bigcup_{j=1}^{t''} F(\mathbf{y}_{i,j}) \middle| \mathbf{y}^* \in \mathsf{V}\right] \geqslant 1 - \kappa^{c''}.$$

for random vectors $\mathbf{y}_{i,j} \leftarrow \mathsf{U}$.

Now set c', c'' such that $\kappa^{-c} \leqslant \kappa^{-c'} + t' \cdot \kappa^{-c''}$ for all sufficiently large κ and $t = t' \cdot t''$. □

G Multi-Input Quadratic FE

In this section, we define multi-input Quadratic Functional Encryption (QFE) schemes and show that even a very basic version of this notion — one that only guarantees single-corruption security and a limited form of correctness — is already sufficient to construct full-fledged FH-IPFE. Thus, our impossibility results also rule out such QFE schemes, highlighting that even bounded corruptions become non-trivial in the multi-input FE setting.

We now define the notion below, following the terminology of [14].

Definition G.1 (Two-input QFE). A two-input private-key QFE scheme $\mathrm{QFE}.\mathcal{E} = (\mathrm{QFE.Setup}, \mathrm{QFE.KGen}, \mathrm{QFE.Enc}, \mathrm{QFE.Dec})$ for $\mathbb{Z}_q^n$ is given by the following PPT algorithms.

- $\mathrm{QFE.Setup}(1^\kappa)$: On input a unary encoded security parameter, it outputs a master secret key msk.
- $\mathrm{QFE.KGen}(\mathsf{msk}, \mathbf{M})$: On input a master secret key msk and an $n \times n$ matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times n}$, the key generation algorithm outputs a secret key $\mathsf{sk}[\mathbf{M}]$.
- $\mathrm{QFE.Enc}(\mathsf{msk}, i \in \{1,2\}, \mathbf{x})$: On input a master secret key msk, a slot index $i \in \{1,2\}$ and a (slotted) plaintext vector $\mathbf{x} \in \mathbb{Z}_q^n$, the encryption algorithm outputs a ciphertext $\mathsf{ct}[i, \mathbf{x}]$.
- $\mathrm{QFE.Dec}(\mathsf{sk}[\mathbf{M}], (\mathsf{ct}[1,\mathbf{x}], \mathsf{ct}[2,\mathbf{y}]))$: On input a secret key $\mathsf{sk}[\mathbf{M}]$ and two slotted ciphertexts $\mathsf{ct}[1,\mathbf{x}]$ and $\mathsf{ct}[2,\mathbf{y}]$, the decryption algorithm outputs $z \in \mathbb{Z}_q$.

We now define the notion of correctness.

- **ν-correctness**: We say $\mathcal{E}$ is ν-correct if the following experiment outputs one with probability at least ν. Sample $\mathsf{msk} \leftarrow \mathrm{QFE.Setup}(1^\kappa)$, $\mathbf{M} \leftarrow \mathbb{Z}_q^{n \times n}$, $\mathsf{sk}[\mathbf{M}] \leftarrow \mathrm{QFE.KGen}(\mathsf{msk}, \mathbf{M})$, $\mathbf{x}$, $\mathbf{y} \leftarrow \mathbb{Z}_q^n$, $\mathsf{ct}[1,\mathbf{x}] \leftarrow \mathrm{QFE.Enc}(\mathsf{msk}, 1, \mathbf{x})$ and $\mathsf{ct}[2,\mathbf{y}] \leftarrow \mathrm{QFE.Enc}(\mathsf{msk}, 2, \mathbf{y})$. The experiment outputs one if $\mathrm{QFE.Dec}(\mathsf{sk}[\mathbf{M}], (\mathsf{ct}[1,\mathbf{x}], \mathsf{ct}[2,\mathbf{y}])) = \mathbf{x}\mathbf{M}\mathbf{y}^T$. We say the scheme satisfies limited correctness if the above condition is only required to hold when $\mathbf{M} = \mathbf{I}$ (the identity matrix), in which case the output is simply the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$.

Remark G.2 Our definition of 2-input FE only allows computing quadratic functions where each quadratic term consists of one variable from each of the two inputs. Naturally, our impossibility result also rules out all stronger variants, for example where quadratic terms can consist of two variables from the same input.

Single-corruption security. A (multi-input) QFE, just as in, FE allows, in general, for many secret-key corruptions. Here we give a very weak definition in which only the secret key for the identity matrix $\mathbf{M} := \mathbf{I}$ is allowed to be corrupted. The notion is defined as follows. Any *admissible* (defined next) PPT adversary $\mathbf{A}$ should have at most a $\frac{1}{2} + \frac{1}{\mathsf{poly}}$ advantage in the following game. The

adversary receives $\mathsf{sk}[\mathbf{I}]$ sampled as in Definition G.1 and may make slotted LoR queries $(i, \mathbf{x}_0, \mathbf{x}_1)$, for arbitrary i, $\mathbf{x}_0$, $\mathbf{x}_1$, replied to with $\mathrm{QFE.Enc}(\mathsf{msk}, i, \mathbf{x}_b)$, where b is the challenge bit. We say $\mathbf{A}$ is admissible if $\mathcal{A}$ never makes two queries $(1, \mathbf{x}_0, \mathbf{x}_1)$ and $(2, \mathbf{x}'_0, \mathbf{x}'_1)$ such that $\langle \mathbf{x}_0, \mathbf{x}'_0 \rangle \neq \langle \mathbf{x}_1, \mathbf{x}'_1 \rangle$.

Two-input single-corruption QFE implies FH-IPFE. We now sketch how to build an FH-IPFE from any two-input single-corruption QFE that yields only limited correctness (Definition G.1).

- $\mathrm{Setup}(1^\kappa)$: run $\mathsf{msk} \leftarrow \mathrm{QFE.Setup}(1^\kappa)$ and $\mathsf{sk} \leftarrow \mathrm{QFE.KGen}(\mathsf{msk}, \mathbf{I})$, where $\mathbf{I}$ is the identity $n \times n$ matrix. Let the new master secret key be $(\mathsf{msk}, \mathsf{sk})$.
- $\mathrm{KGen}((\mathsf{msk}, \mathsf{sk}), \mathbf{u})$: output $(\mathsf{sk}, \mathsf{ct}[1, \mathbf{u}])$, where $\mathsf{ct}[1, \mathbf{u}] \leftarrow \mathrm{QFE.Enc}(\mathsf{msk}, 1, \mathbf{u})$.
- $\mathrm{Enc}((\mathsf{msk}, \mathsf{sk}), \mathbf{x})$: output $\mathsf{ct}[2, \mathbf{x}] \leftarrow \mathrm{QFE.Enc}(\mathsf{msk}, 2, \mathbf{x})$.
- $\mathrm{Dec}((\mathsf{sk}, \mathsf{ct}[1, \mathbf{u}]), \mathsf{ct}[2, \mathbf{x}])$: output $\mathrm{QFE.Dec}(\mathsf{sk}, (\mathsf{ct}[1, \mathbf{u}]), \mathsf{ct}[2, \mathbf{x}]))$.

The resulting IPFE is $\nu-$correct if the base two-input QFE is $\nu-$limited correct. Moreover, the resulting IPFE is FH-IND-CPA secure (Definition D.1) if the base QFE is single-corruption secure.