Adaptive Simulation Security for Inner Product Functional Encryption

Shweta Agrawal[1], Benoît Libert[2,3], Monosij Maitra[1], and Radu Titiu[3,4]

[1] IIT Madras, India
[2] CNRS, Laboratoire LIP, France
[3] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), France
[4] Bitdefender, Bucharest, Romania

Abstract. Inner product functional encryption (IPFE) [1] is a popular primitive which enables inner product computations on encrypted data. In IPFE, the ciphertext is associated with a vector $\boldsymbol{x}$, the secret key is associated with a vector $\boldsymbol{y}$ and decryption reveals the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Previously, it was known how to achieve adaptive *indistinguishability* (IND) based security for IPFE from the DDH, DCR and LWE assumptions [8]. However, in the stronger simulation (SIM) based security game, it was only known how to support a restricted adversary that makes all its key requests either before or after seeing the challenge ciphertext, but not both. In more detail, Wee [46] showed that the DDH-based scheme of Agrawal *et al.* (Crypto 2016) achieves *semi-adaptive* simulation-based security, where the adversary must make all its key requests *after* seeing the challenge ciphertext. On the other hand, O'Neill showed that all IND-secure IPFE schemes (which may be based on DDH, DCR and LWE) satisfy SIM based security in the restricted model where the adversary makes all its key requests *before* seeing the challenge ciphertext.

In this work, we resolve the question of SIM-based security for IPFE by showing that variants of the IPFE constructions by Agrawal *et al.*, based on DDH, Paillier and LWE, satisfy the strongest possible *adaptive* SIM-based security where the adversary can make an unbounded number of key requests both before and after seeing the (single) challenge ciphertext. This establishes optimal security of the IPFE schemes, under all hardness assumptions on which it can (presently) be based.

Keywords. Functional encryption, inner-products, simulation-based security, standard assumptions.

1 Introduction

Functional Encryption (FE) [15, 37] is a modern cryptographic paradigm that allows fine-grained access to encrypted data, unlike traditional public-key encryption, where decryption offers all-or-nothing access to data. In FE, a secret key sk_f corresponds to function f, and ciphertext $\mathsf{ct}(\boldsymbol{x})$ corresponds to some input $\boldsymbol{x}$ from the domain of f. Given a function key sk_f and a ciphertext $\mathsf{ct}(\boldsymbol{x})$, a user can run the decryption algorithm to learn $f(\boldsymbol{x})$. Security of FE guarantees that beyond $f(\boldsymbol{x})$, nothing about $\boldsymbol{x}$ is revealed.

Functional encryption has been studied extensively, yielding a plethora of constructions that achieve various tradeoffs between generality, security and hardness assumptions. Assuming the existence of the powerful multilinear maps [22] or indistinguishability obfuscation [23], FE can be constructed for all polynomial sized circuits achieving the strongest possible definition of security [23, 24]. However, from standard assumptions, which is the focus of this work, constructions are only known for restricted classes of functionalities or achieving restricted notions of security. We discuss each of these aspects next.

On the Definition of Security. In the papers that introduced functional encryption [15, 37], the authors discussed the subtleties involved in formulating the *right* definition of security for FE. Traditionally, an "indistinguishability" (IND) style definition had been used for constructing various special cases of functional encryption, which roughly requires that no efficient adversary that has oracle access to the key generation algorithm should be able to distinguish between encryptions of two messages $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. However, [15] showed that this notion was too weak for functional encryption in some cases. Specifically, they gave an FE construction that could be proved secure with respect to the IND security requirement, but was intuitively insecure.

[15, 37] proposed the study of *simulation-based* (SIM) security which asks that the view of the adversary be simulated by a simulator that is given access to pairs $(f_i, f_i(\boldsymbol{x}^\star))$ where f_i are the functions for which the adversary requests keys, and $\boldsymbol{x}^\star$ is the challenge message. SIM security captured the intuition that nothing about $\boldsymbol{x}^\star$ be revealed except for the function output value, and ruled out the insecure scheme that IND security could not. However, it was soon shown that for general functionalities, SIM-based security is impossible to achieve [15, 7].

Additionally, other restricted notions of security have also been studied, that limit either i) the number of key requests – *bounded collusion* FE [26], ii) the "type" of key requests – *one sided FE or "predicate encryption"* where the adversary may only request keys for functions f such that $f(\boldsymbol{x}^\star) = 0$ [28, 10], or iii) that allow for part of the input vector to be public – *public index or "attribute-based encryption"* [29, 31, 6, 14, 28]. While these restricted notions are meaningful for different applications, it remains desirable to obtain security in an unrestricted security game, if only for specialized functionalities.

Restricting the Functionality. Aside from different security notions, constructions of FE also vary in the functionality they support. Many special cases of FE have been studied before and since its formalization as an abstract primitive [15, 37] – identity-based encryption (IBE) [13], [42] fuzzy identity-based encryption [41], [5] attribute-based encryption (ABE) [29, 31, 27] predicate encryption (PE) [31, 30, 28], bounded-key functional encryption [26, 25]. However, excepting [30], the security of all these schemes was restricted in one of the three ways discussed above.

Abdalla, Bourse, De Caro and Pointcheval [1] introduced the primitive of inner product functional encryption (IPFE). In IPFE the ciphertext is associated with a vector $\boldsymbol{x}$, the secret key is associated with a vector $\boldsymbol{y}$ and decryption

reveals $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Since its introduction, IPFE has been studied extensively [1, 12, 19, 8, 43, 11, 17, 44] due to its feasibility under well-established assumptions [1, 8], its natural applications [1] and extensions [3, 21, 2, 18], its use as a building block for more advanced functionalities [33, 3, 32, 4, 9], and the fact that it admits an unrestricted security definition (more on this below).

Security of IPFE. Abdalla et al. [1] constructed practical schemes for IPFE under well studied hardness assumptions like the Decisional Diffie-Hellman (DDH) and Learning With Errors (LWE). Their constructions achieved security in a game which did not place any restriction on the number or type of key requests, nor necessitated making any part of the input public. Given the paucity of schemes that achieve these features, this was good news.

However, despite its positive features, the security game considered by [1] had shortcomings – their constructions were only proven to be *selectively secure* in the IND model, which means that the adversary has to announce the challenge messages before it even sees the public key of the scheme. This result was improved by Agrawal, Libert and Stehlé [8] who constructed adaptive AD-IND functional encryption for the same inner product functionality, under DDH , LWE and also from Paillier's Decision Composite Residuosity (DCR). Thus, the result of [8] established optimal security of IPFE in the IND-based game, from all hardness assumptions on which it can (presently) be based.

In the domain of SIM-based security for IPFE, much less is known. On one hand, O'Neill [37] showed that for IPFE,[5] IND security implies SIM security in a model where the adversary is restricted to making all its key queries *before* it sees the challenge ciphertext. On the other hand, Wee [46] recently proved that the DDH-based FE scheme from [8] achieves simulation-based security in a model where the adversary is restricted to making all its key queries *after* it sees the challenge ciphertext, in the so-called *semi-adaptive* game. Datta *et al.* [20] subsequently extended Wee's ideas so as to prove simulation-security against adaptive adversaries in predicate encryption schemes [30] based on bilinear maps [34–36]. In the IPFE setting, known proofs of SIM security break down in the natural *adaptive* model where the adversary is allowed to make key queries adaptively, both before and after seeing the challenge ciphertext. Moreover, Wee's result is not generic and only applies to the DDH-based construction of [8] as well as in specific pairing-based constructions of predicate encryption.

For a functionality as basic as IPFE, this state of affairs is quite dissatisfying. Specifically, the following fundamental question remains to be answered:

Is it possible to achieve the strongest notion of security, namely AD-SIM *security for* IPFE*, which permits the adversary an unbounded number of key requests before and after seeing the (single) challenge ciphertext? Moreover, can we achieve* AD-SIM *security from all the assumptions on which* IPFE *can be based, namely* DDH *(in groups without a bilinear map),* DCR *and* LWE*?*

In the present work, we resolve this question in the affirmative.

[5] Or, more generally, the class of *preimage sampleable functinalities* of which inner product is a special case.

Our Results. In this work, we prove adaptive simulation-security (AD-SIM) for an unbounded number of key queries and a single challenge ciphertext, for IPFE schemes, based on the DDH, DCR and LWE assumptions. We place no restrictions on when the adversary may query for keys with respect to the challenge ciphertext. Thus, our security game achieves the "best of" both security games considered by Wee [46] and O'Neill [37], where the former permits post-challenge key requests but not pre, and the latter permits pre-challenge key requests but not post. By providing constructions under all assumptions on which IPFE schemes may presently be based, we improve a result by Wee [46], which achieved semi-adaptive SIM based security for DDH-based IPFE.

In more detail, we prove that the DDH based scheme of Agrawal et al. [8] (unmodified) achieves AD-SIM rather than just AD-IND security. Next, we show how to modify the DCR based scheme of [8] so that it satisfies AD-SIM security. Finally, we construct a new scheme for IPFE mod p based on LWE which leverages the LWE scheme of [8] (almost) generically to achieve AD-SIM security. Note that the impossibility from [15] rules out AD-SIM for many challenge messages, but our proofs work for a single challenge message (as does [46]). Moreover, [7] shows that AD-SIM security for one challenge message is impossible for *all* circuits, but this does not contradict our results since our proofs apply for a restricted class of functionality. Since our schemes achieve the strongest possible SIM security notion for IPFE under all assumptions on which it can currently be based, we finally settle the question of optimal security for IPFE.

Technical overview. Next, we provide a technical overview of our constructions in turn.

DDH-based IPFE: The DDH-based IPFE scheme of Agrawal *et al.* [8] was shown to provide indistinguishability-based security against adaptive adversaries (or AD-IND security for short). Later on, Abdalla *et al.* [3] proved it simulation-secure against selective adversaries. Wee [46] subsequently gave a proof of semi-adaptive simulation-based security for the same construction. Here, we show that the scheme can actually be proved simulation-secure against adaptive adversaries without any modification.

In Wee's proof [46], the simulator can create a dummy challenge ciphertext as an encryption of the all-zeroes vector. In the semi-adaptive setting, the simulated challenge ciphertext does not have to be consistent with pre-challenge queries because functional key queries are only allowed *after* the challenge phase. For a post-challenge key query $\boldsymbol{y} \in \mathbb{Z}_q^\ell$, the simulator has to respond with a key that decrypts the dummy ciphertext to the value $z_{\boldsymbol{y}} = f_{\boldsymbol{y}}(\boldsymbol{x}^\star) = \langle \boldsymbol{y}, \boldsymbol{x}^\star \rangle$ supplied by the oracle. To do this, it can embed the value $z_{\boldsymbol{y}} = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$ in the modified secret key which is obtained as an appropriate shift of the actual secret key. Namely, if the master public key is $g^{\boldsymbol{s}} \cdot h^{\boldsymbol{t}} \in \mathbb{G}^\ell$ and the master secret key consists of $(\boldsymbol{s}, \boldsymbol{t}) \in_R \mathbb{Z}_q^\ell \times \mathbb{Z}_q^\ell$, the real functional secret key for $\boldsymbol{y} \in \mathbb{Z}_q^\ell$ is comprised of $(s_{\boldsymbol{y}}, t_{\boldsymbol{y}}) = (\langle \boldsymbol{s}, \boldsymbol{y} \rangle, \langle \boldsymbol{t}, \boldsymbol{y} \rangle)$. In order to "program" $z_{\boldsymbol{y}} = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$ in the simulated post-challenge keys, the simulator can define

$$s'_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \alpha \cdot z_{\boldsymbol{y}} \bmod q \qquad t'_{\boldsymbol{y}} := \langle \boldsymbol{t}, \boldsymbol{y} \rangle + \beta \cdot z_{\boldsymbol{y}} \bmod q,$$

for carefully chosen coefficients $\alpha, \beta \in \mathbb{Z}_q$. From the adversary's view, this is equivalent to changing the master secret key into $\boldsymbol{s}' = \boldsymbol{s} + \alpha \cdot \boldsymbol{x}^\star \bmod q$ and $\boldsymbol{t}' = \boldsymbol{t} + \beta \cdot \boldsymbol{x}^\star \bmod q$, which is consistent with the master public key and reponses to key queries for all vectors $\boldsymbol{y}$. Using a careful analysis, it was shown [46] that, under the DDH assumption, the simulation is indistinguishable from the real experiment, even if the message $\boldsymbol{x}^\star$ is adaptively chosen after seeing the public parameters, but before making any key query.

In order to prove simulation-based security for adaptive adversaries, we use the same approach as [46], but we modify the generation of the simulated ciphertext. Now, the dummy ciphertext should not only decrypt to the values dictated by the oracle under post-challenge keys, but it also needs to be consistent with responses to pre-challenge queries. To achieve this, our simulator answers pre-challenge key queries by running the real functional key generation algorithm. For each key query $\boldsymbol{y} \in \mathbb{Z}_q^\ell$, it replies with $(\boldsymbol{s}_{\boldsymbol{y}}, \boldsymbol{t}_{\boldsymbol{y}}) = (\langle \boldsymbol{s}, \boldsymbol{y} \rangle, \langle \boldsymbol{t}, \boldsymbol{y} \rangle)$. In the challenge phase, the simulator has to create a ciphertext that is compatible with all the pre-challenge queries without having access to the challenge message $\boldsymbol{x}^\star \in \mathbb{Z}_q^\ell$. For this purpose, it encrypts an arbitrary dummy message $\bar{\boldsymbol{x}}$ that satisfies the relations $\langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle \bmod q$, for any pre-challenge query $\boldsymbol{y} \in \mathbb{Z}_q^\ell$. Our observation is that, although the DDH-based scheme of [8] encrypts vectors $\boldsymbol{x} \in \mathbb{Z}^\ell$ with small entries (because functional secret keys only make it possible to recover the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ when it lives in a polynomial-size interval), the dummy message does not have to be small. This implies that, given the function evaluation $\{z_{\boldsymbol{y}} = f_{\boldsymbol{y}}(\boldsymbol{x}^\star) = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle\}_{\boldsymbol{y}}$ corresponding to all pre-challenge queries $\boldsymbol{y}$, the simulator can easily compute a compatible dummy message using linear algebra over $\mathbb{Z}_q$. Once the simulator is committed to the challenge ciphertext, it has to "program" the post-challenge functional keys in such a way that they decrypt the dummy ciphertext to the real function evaluations $z_{\boldsymbol{y}} = f_{\boldsymbol{y}}(\boldsymbol{x}^\star) = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$. Given a post-challenge query $\boldsymbol{y} \in \mathbb{Z}_q^\ell$ and the corresponding function evaluation $z_{\boldsymbol{y}} = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$, the value $z_{\boldsymbol{y}}$ is embedded in the simulated functional key in such a way that the difference $z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle$ between $z_{\boldsymbol{y}}$ and the function evaluation $f_{\boldsymbol{y}}(\bar{\boldsymbol{x}})$ serves as a shift of the real actual key: namely, the simulator returns $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$, where

$$\begin{aligned} s'_{\boldsymbol{y}} &:= \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle) \bmod q \\ t'_{\boldsymbol{y}} &:= \langle \boldsymbol{t}, \boldsymbol{y} \rangle + \beta \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle) \bmod q. \end{aligned} \tag{1.1}$$

By exploiting the linearity properties of the scheme, the shift terms $\alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle)$ and $\beta \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle)$ ensure that $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$ will decrypt the dummy ciphertext to the oracle-supplied $z_{\boldsymbol{y}}$. As in [46], we can prove that this shift of post-challenge keys is equivalent to a shift of the master secret key from the adversary's view: namely, $\mathsf{msk} = (\boldsymbol{s}, \boldsymbol{t})$ is traded for $\mathsf{msk}' = (\boldsymbol{s}', \boldsymbol{t}')$, where $\boldsymbol{s}' = \boldsymbol{s} + \alpha \cdot (\boldsymbol{x}^\star - \bar{\boldsymbol{x}})$ and $\boldsymbol{t}' = \boldsymbol{t} + \beta \cdot (\boldsymbol{x}^\star - \bar{\boldsymbol{x}})$. By applying complexity leveraging argument in a statistical setting (as previously done in, e.g., [45, 11, 46]), we can prove that the two master secret keys of $(\boldsymbol{s}', \boldsymbol{t}')$ and $(\boldsymbol{s}, \boldsymbol{t})$ are identically distributed in the adversary's view, even if the adversary chooses $\boldsymbol{x}^\star$ adaptively, after having seen

the public parameters and responses to pre-challenge queries.

DCR-based IPFE: The above ideas can be adapted to the Composite Residuosity assumption (DCR) [38] so as to prove simulation-based security in (a variant of) the Paillier-based construction of Agrawal *et al.* [8]. One difficulty is that functional secret keys $\boldsymbol{s}_{\boldsymbol{y}} = \langle \boldsymbol{s}, \boldsymbol{y} \rangle$ have to be computed over the integers since the group order is hidden. When we want to prove that the simulation is indistinguishable from the real experiment, this makes it harder to create simulated functional secret keys $\boldsymbol{s}'_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle)$ that are statistically indistinguishable from the real keys $\boldsymbol{s}_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle$. In particular, since the functional secret keys are computed over $\mathbb{Z}$, the simulator cannot easily compute a small-norm dummy message $\bar{\boldsymbol{x}}$ which is consistent with responses to pre-challenge queries (indeed, it does not have a short basis for the lattice induced by these queries). However, the simulator can still use the pre-challenge queries to compute a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ with large entries. Although $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ does not fit in $\mathbb{Z}_N^\ell$, we can still encrypt $\bar{\boldsymbol{x}}$ mod N and obtain a simulated ciphertext which is compatible with responses to pre-challenge queries. When it comes to simulating post-challenge keys, we can have the simulator compute $\boldsymbol{s}'_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle)$ and argue that, from the adversary's view, this is equivalent to trading the master secret key $\boldsymbol{s} \in \mathbb{Z}^\ell$ for $\boldsymbol{s}' := \boldsymbol{s} + \alpha \cdot (\boldsymbol{x}^\star - \bar{\boldsymbol{x}})$. By computing an upper bound for $\|\boldsymbol{x}^\star - \bar{\boldsymbol{x}}\|_\infty$, we can increase the magnitude of the master secret key $\boldsymbol{s} \in \mathbb{Z}^\ell$ so as to make sure that the statistical distance between $\boldsymbol{s} \in \mathbb{Z}^\ell$ and $\boldsymbol{s}' \in \mathbb{Z}^\ell$ negligible. This is actually possible by sampling the entries of the master secret key $\boldsymbol{s} \in \mathbb{Z}^\ell$ from a large interval, so that its bitlength becomes $O(\ell^3 \cdot \lambda^3 / \mathrm{polylog}(\lambda))$ if ℓ is the dimension of encrypted vectors.

LWE-based IPFE mod p: We now outline our adaptation of the LWE-based construction for IPFE [8] to achieve adaptive SIM-based security. We focus on the construction of IPFE modulo a prime p, [8, Sec 4.2], where the ciphertext contains a vector $\boldsymbol{x} \in \mathbb{Z}_p^\ell$, the key contains a vector $\boldsymbol{y} \in \mathbb{Z}_p^\ell$ and decryption reveals $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ mod p. Our construction is generic except that it requires the underlying scheme IPFE to satisfy the property that functional keys for vectors that are linearly dependent on previously queried vectors may be computed as the linear combination of previously returned keys. In more detail, say that the adversary queries vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k \in \mathbb{Z}_p^\ell$ and then submits a query $\boldsymbol{y}$ such that $\boldsymbol{y} = \sum_{j \in [k]} k_j \cdot \boldsymbol{y}_j \pmod p$, for some $k_j \in \mathbb{Z}_p$. Then, the secret key $\mathsf{sk}_{\boldsymbol{y}} \in \mathbb{Z}^m$ can be computed as $\mathsf{sk}_{\boldsymbol{y}} = \sum_{j \in [k]} k_j \cdot \mathsf{sk}_{\boldsymbol{y}_j} \in \mathbb{Z}^m$. This property is satisfied by the LWE-based construction that evaluates inner products over $\mathbb{Z}_p$ in [8, Sec 4.2].

Since secret keys of linearly dependent vectors are computed as linear combinations of previously returned keys[6], it suffices to consider an adversary that only requests for $\ell - 1$ linearly independent keys. Let us refer to the key requests made in the pre-challenge phase as $\boldsymbol{y}^{\mathsf{pre}}$ and those in the post-challenge phase as $\boldsymbol{y}^{\mathsf{post}}$.

[6] As in [8], this results in a stateful key generator.

To begin, we set $L = 2\ell$ and instantiate the adaptive IND secure IPFE of [8, Sec 4.2] with message length L. Given a message vector $\boldsymbol{x} \in \mathbb{Z}_p^\ell$, we extend it to $\widehat{\boldsymbol{x}} \in \mathbb{Z}_p^L$ to make one slot corresponding to each independent key queried up to $\ell - 1$ keys. The simulated challenge ciphertext $\boldsymbol{c}^*$ encrypts the extended vector $\widehat{\bar{\boldsymbol{x}}} = (\bar{\boldsymbol{x}}, -r_1, \ldots, -r_{\ell-1}, 1)$ for a dummy message $\bar{\boldsymbol{x}}$, where $\{r_i \leftarrow \mathbb{Z}_p\}_{i\in[\ell-1]}$ are chosen uniformly at random, while simulating the setup phase of the real protocol.

Pre-challenge keys for vectors $\boldsymbol{y}^{\mathsf{pre}} \in \mathbb{Z}_p^\ell$ are handled as in the real scheme. In more detail, for the ith independent pre-challenge key $\boldsymbol{y}_i^{\mathsf{pre}}$ the underlying IPFE scheme is used to compute keys for the vector $(\boldsymbol{y}_i^{\mathsf{pre}}, \mathbf{e}_i, r_i)$, where $\mathbf{e}_i \in \mathbb{Z}_p^{\ell-1}$ is the i-th canonical vector and $\{r_i \leftarrow \mathbb{Z}_p\}_{i\in[\ell-1]}$ are as above. The challenge ciphertext is handled by computing a message $\bar{\boldsymbol{x}}$ that is consistent with only the keys associated with the pre-challenge vectors $\boldsymbol{y}^{\mathsf{pre}}$. For handling post-challenge queries, let $\Delta_i = \langle \boldsymbol{x}^\star - \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle$ be the difference in decryption using the i-th post-challenge key corresponding to a linearly independent vector $\boldsymbol{y}_i^{\mathsf{post}}$. To compensate this difference, we extend the vector $\boldsymbol{y}_i^{\mathsf{post}}$ to $(\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, \Delta_i + r_i)$. Note that the randomizer r_i in the i-th slot of the extended message vector hides Δ_i in the i-th post-challenge key.

For post challenge keys, the decryption outputs

$$\langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle + \langle -r_i, 1 \rangle + \langle 1, \Delta_i + r_i \rangle = \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle + \Delta_i = \langle \boldsymbol{x}^\star, \boldsymbol{y}_i^{\mathsf{post}} \rangle$$

as desired. It is easy to verify that this also works if a post-challenge vector $\boldsymbol{y}^{\mathsf{post}}$ is a linear combination of possibly any arbitrary subset of pre-challenge and post-challenge keys queried so far. As for pre-challenge queries, the simulated keys properly decrypt the simulated ciphertext since the r_i simply get cancelled (i.e., $\Delta_i = 0$). Also, note that the simulated keys work for any honestly generated ciphertext since this contains $\mathbf{0}$ in the extended slots and do not "activate" the extended slots in the keys. For the detailed proof, please see Section 5.

DCR-based IPFE mod N: In the description of our construction from LWE, we assumed that the modulus p is prime. However, the same technique can also be applied to the Paillier-based construction of [8, Section 5.2], which evaluates inner products over $\mathbb{Z}_N$. As a result, it provides a simulation-secure IPFE with stateful key generation for inner products over $\mathbb{Z}_N$, whereas our scheme in Section 4 is stateless but computes inner products over $\mathbb{Z}$. When we switch to composite moduli $N = pq$, we need to take into account that $\mathbb{Z}_N$ is not a field when the simulator has to solve a linear system over $\mathbb{Z}_N$ in order to compute a dummy message. Fortunately, inversion over $\mathbb{Z}_N$ is always possible with overwhelming probability when factoring N is hard.

2 Preliminaries

In this section we define the preliminaries that we require in this work.

Notation. We begin by defining the notation that we will use throughout the paper. We use bold letters to denote vectors and the notation $[1, n]$ or $[n]$ or

$\{1, \ldots, n\}$ interchangeably to denote the set of first n positive integers. We denote by $U([n])$ the uniform distribution over the set $[n]$ and $u \hookleftarrow \mathcal{D}$ or $u \leftarrow \mathcal{D}$ interchageably to sample an element u from distribution $\mathcal{D}$. Concatenation is denoted by the symbol $\|$ or $|$ interchangeably. We say a function $f(n)$ is *negligible*, denoted by $\mathrm{negl}(n)$, if it is $O(n^{-c})$ for all $c > 0$. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$.

2.1 Useful Lemmas

We will rely on a few simple but useful lemmas, which are stated hereunder.

Lemma 1. *Let M, m be positive integers, $M = m \cdot q + r$ with $0 \leq r < m$. The statistical distance between the distributions $(U(\mathbb{Z}_M) \bmod m)$ and $U(\mathbb{Z}_m)$ is bounded by $\Delta(U(\mathbb{Z}_M) \bmod m, U(\mathbb{Z}_m)) \leq \frac{r}{M}$.*

Proof. Let $M = mq + r$, with $0 \leq r < m$. Observe that for $i \in \mathbb{Z}_m$ we can compute the number of integers of the form $i + jm$, smaller than $M - 1$, by $\lfloor \frac{M-1-i}{m} \rfloor + 1$ which is also equal to $\lfloor q + \frac{r-1-i}{m} \rfloor + 1$. So the probability of getting $i \in \mathbb{Z}_m$ by sampling from $U(\mathbb{Z}_M) \mod m$ is equal to $\frac{q+1}{M}$ if $i < r$ or equal to $\frac{q}{M}$ if $i \geq r$. So the statistical distance that we want to evaluate is equal to:

$$\Delta = \frac{1}{2}\left(\sum_{i<r}\left|\frac{q+1}{M} - \frac{1}{m}\right| + \sum_{i\geq r}\left|\frac{q}{M} - \frac{1}{m}\right|\right) = \frac{r(m-r)}{Mm} \leq \frac{r}{M}.$$

□

Lemma 2. *Let $a, b, c \in \mathbb{Z}$ such that $b > a$. We have $\Delta\left(U_{[a,b]}, U_{c+[a,b]}\right) \leq \frac{|c|}{b-a}$, where $U_{[\alpha,\beta]}$ is the uniform distribution on $[\alpha, \beta] \cap \mathbb{Z}$.*

Lemma 3. *For any $\mathbf{A} \in \mathbb{R}^{m\times n}$, let $\alpha := \max_{i,j} |a_{i,j}|$. Then, we have the inequality $\det(\mathbf{A}\mathbf{A}^\top) \leq (n \cdot \alpha^2)^m$.*

Proof. Since $\mathbf{A}\mathbf{A}^\top \in \mathbb{R}^{m\times m}$ is positive definite, we know that it has positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m \geq 0$. By the mean inequality, we have $\sqrt[m]{\lambda_1\lambda_2\cdots\lambda_m} \leq \frac{\lambda_1+\cdots+\lambda_m}{m}$. This can be interpreted as $\det(\mathbf{A}\mathbf{A}^\top) \leq \left(\frac{\mathsf{Tr}\mathbf{A}\mathbf{A}^\top}{m}\right)^m$ and the right hand side term can be bounded by $(n\alpha^2)^m$. □

Lemma 4. *Let $\mathbf{Y} \in \mathbb{Z}^{k\times\ell}$ be a full rank matrix such that $\max_{i,j} |y_{ij}| \leq Y$. There exists an efficient algorithm that finds a basis $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\ell-k}\} \subset \mathbb{Z}^\ell$ of the lattice $\mathbf{Y}^\perp := \{\mathbf{x} \in \mathbb{Z}^\ell : \mathbf{Y} \cdot \mathbf{x} = \mathbf{0}\}$ such that*

$$\|\mathbf{x}_j\|_\infty \leq (\sqrt{k}Y)^k \ , \ j \in \{1, \ldots, \ell - k\}.$$

Proof. We assume w.l.o.g. that $\mathbf{Y} = [\mathbf{A}|\mathbf{B}]$, for a full rank matrix $\mathbf{A} \in \mathbb{Z}^{k\times k}$ and for some $\mathbf{B} \in \mathbb{Z}^{k\times(\ell-k)}$ such that $\max_{i,j} |a_{i,j}| \leq Y$ and $\max_{i,j} |b_{i,j}| \leq Y$. If $\mathbf{x} = (z_1, \ldots, z_k, \lambda_1, \ldots, \lambda_{\ell-k})^\top$ satisfies $\mathbf{Y} \cdot \mathbf{x} = \mathbf{0}$, Cramer's rule implies

$$z_i = \frac{-1}{\det \mathbf{A}} \sum_{j=1}^{\ell-k} \lambda_j \cdot \det \mathbf{A}_{ij},$$

where the matrix $\mathbf{A}_{ij} \in \mathbb{Z}^{k \times k}$ is obtained by replacing the i-th column of $\mathbf{A}$ by the j-th column of $\mathbf{B}$. By choosing $(\lambda_1, \lambda_2, \ldots, \lambda_{\ell-k}) \in \mathbb{Z}^{\ell-k}$ from the set

$$\{\det \mathbf{A} \cdot (1, 0, \ldots, 0), \det \mathbf{A} \cdot (0, 1, \ldots, 0), \ldots, \det \mathbf{A} \cdot (0, 0, \ldots, 1)\},$$

we obtain the desired basis. Concretly, for every $j \in \{1, 2, \ldots, \ell - k\}$, we define

$$\mathbf{x}_j = (-\det \mathbf{A}_{1j}, -\det \mathbf{A}_{2j}, \ldots, -\det \mathbf{A}_{kj},\ \mathbf{e}_j \cdot \det \mathbf{A}) \in \mathbb{Z}^{\ell}.$$

By using Lemma 3 we get the bounds on the size of each basis vector $\mathbf{x}_j$. □

Corollary 1. *Let a full rank $\mathbf{Y} \in \mathbb{Z}^{k \times \ell}$ such that $|y_{ij}| \leq Y$ and $\mathbf{z} \in \mathbb{Z}^k$. If there exists a solution $\mathbf{x}_0 \in \mathbb{Z}^{\ell}$ to the system $\mathbf{Y} \cdot \mathbf{x}_0 = \mathbf{z}$, then there exists an efficient algorithm that computes a solution $\mathbf{x} \in \mathbb{Z}^{\ell}$ such that $\|\mathbf{x}\|_\infty \leq (\ell - k) \cdot (\sqrt{k}Y)^k$.*

Proof. By Lemma 4, we can efficiently find a basis $\{\mathbf{x_1}, \ldots, \mathbf{x}_{\ell-k}\}$ of the lattice $\mathbf{Y}^{\perp}$ such that $\|\mathbf{x}_j\|_\infty \leq (\sqrt{k}Y)^k$. Reducing the solution $\mathbf{x}_0$ modulo this basis, we obtain $\mathbf{x} := \mathbf{x}_0 \bmod \mathbf{Y}^{\perp}$ such that $\|\mathbf{x}\|_\infty \leq \sum_{k=1}^{\ell-k} \|\mathbf{x}_j\|_\infty \leq (\ell - k) \cdot (\sqrt{k}Y)^k$. □

2.2 Functional Encryption

Definition 1. *A Functional Encryption (FE) scheme over a class of functions $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Z}\}$ consists of the* PPT *algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$*:*

$\mathbf{Setup}(1^\lambda, \mathcal{F})$: *Outputs a public key* mpk *and a master secret key* msk.

$\mathbf{Keygen}(\mathsf{msk}, f)$: *Given the master secret key and a functionality $f \in \mathcal{F}$, the algorithm outpus a secret key sk_f.*

$\mathbf{Encrypt}(\mathsf{mpk}, \boldsymbol{x})$: *On input the public key and a message $\boldsymbol{x} \in \mathcal{X}$ from the message space, the algorithm outputs a ciphertext $\boldsymbol{c}$.*

$\mathbf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_f, \boldsymbol{c})$: *Given a ciphertext and a secret key corresponding to some functionality $f \in \mathcal{F}$, the algorithm outputs $\boldsymbol{z} \in \mathcal{Z}$.*

Correctness: We require that for $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$, for all $\boldsymbol{x} \in \mathcal{X}$, all $f \in \mathcal{F}$, $\boldsymbol{c} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x})$ and $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$, with overwhelming probability, we have $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_f, \boldsymbol{c}) = f(\boldsymbol{x})$.

In some cases, we will also give a state st as input to algorithm Keygen, so that a stateful authority may reply to key queries in a way that depends on the queries that have been made so far. In that situation, algorithm Keygen may additionally update state st.

2.3 Security

Next, we define security of functional encryption. Security comes in two flavours – indistinguishability-based and simulation-based – we define each in turn.

Indistinguishability-based security. We first define the weaker notion of indistinguishability-based security [15]. In this notion, one asks that no efficient adversary be able to differentiate encryptions of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ without obtaining secret keys sk_f such that $f(\boldsymbol{x}_0) \neq f(\boldsymbol{x}_1)$.

Definition 2 (Indistinguishability-based security). *A functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *provides semantic security under chosen-plaintext attacks (or IND-CPA security) if no* PPT *adversary has non-negligible advantage in the following game, where* $q_1 \leq q \in \mathsf{poly}(\lambda)$*:*

1. *The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and the master public key* mpk *is given to the adversary* $\mathcal{A}$*.*
2. *The adversary adaptively makes secret key queries to the challenger. At each query, adversary* $\mathcal{A}$ *chooses a function* $f \in \mathcal{F}$ *and obtains* $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$*.*
3. *Adversary* $\mathcal{A}$ *chooses distinct messages* $\boldsymbol{x}_0, \boldsymbol{x}_1$ *subject to the restriction that, if* $\{f_i\}_{i=1}^{q_1}$ *denotes the set of secret key queries made by* $\mathcal{A}$ *at Stage* 2*, it holds that* $f_i(\boldsymbol{x}_0) = f_i(\boldsymbol{x}_1)$ *for each* $i \in \{1, \ldots, q_1\}$*. Then, the challenger flips a fair coin* $\beta \hookleftarrow \{0,1\}$ *and computes* $\boldsymbol{c}^\star \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x}_\beta)$ *which is sent as a challenge to* $\mathcal{A}$*.*
4. *Adversary* $\mathcal{A}$ *makes further secret key queries for arbitrary functions* $f \in \mathcal{F}$*. However, it is required that* $f(\boldsymbol{x}_0) = f(\boldsymbol{x}_1)$ *at each query* $f \in \{f_{q_1+1}, \ldots, f_q\}$*.*
5. *Adversary* $\mathcal{A}$ *eventually outputs a bit* $\beta' \hookleftarrow \{0,1\}$ *and wins if* $\beta' = \beta$*.*

The adversary's advantage is defined to be $\mathbf{Adv}_{\mathcal{A}}(\lambda) := |\Pr[\beta' = \beta] - 1/2|$*, where the probability is taken over all coin tosses.*

Definition 2 captures *adaptive* security in that the adversary is allowed to choose the messages $\boldsymbol{x}_0, \boldsymbol{x}_1$ at Stage 3.

As pointed out in [15], indistinguishability-based security is not fully satisfactory in general as it may fail to rule out constructions that are intuitively insecure. They argue that, whenever it is possible at all, one should prefer a stronger notion of simulation-based security. We recall this notion hereunder.

Simulation-Based Security: For a FE scheme defined as above, a PPT adversary $\mathcal{A} = (A_1, A_2)$ and a PPT simulator $\mathsf{Sim} = (\mathsf{Setup}^\star, \mathsf{KeyGen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{KeyGen}_1^\star)$, consider the following experiments:

$\underline{\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{FE},\mathcal{A}}(1^\lambda)}$	$\underline{\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{FE},\mathcal{A}}(1^\lambda)}$
1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$	1. $(\mathsf{mpk}^\star, \mathsf{msk}^\star) \leftarrow \mathsf{Setup}^\star(1^\lambda, \mathcal{F})$
2. $(\boldsymbol{x}^\star, \mathsf{st}) \leftarrow A_1^{\mathsf{Keygen}(\mathsf{msk}, \cdot)}(\mathsf{mpk})$	2. $(\boldsymbol{x}^\star, \mathsf{st}) \leftarrow A_1^{\mathsf{Keygen}_0^\star(\mathsf{msk}^\star, \cdot)}(\mathsf{mpk}^\star)$ Let $\mathcal{V} = \{(f_i, f_i(\boldsymbol{x}^\star), \mathsf{sk}_{f_i})\}_{i=1}^k$
3. $\boldsymbol{c} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x}^\star)$	3. $(\boldsymbol{c}^\star, \mathsf{st}') \leftarrow \mathsf{Encrypt}^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \mathcal{V}, 1^{\lvert\boldsymbol{x}^\star\rvert})$
4. $\alpha \leftarrow A_2^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}, \boldsymbol{c}, \mathsf{st})$	4. $\alpha \leftarrow A_2^{\mathsf{KeyGen}_1^\star(\mathsf{msk}^\star, \mathsf{st}', \cdot)}(\mathsf{mpk}^\star, \boldsymbol{c}^\star, \mathsf{st})$

In the Ideal experiment above, the $\{f_i \in \mathcal{F}\}_{i=1}^k$ are the functionalities for which the adversary requests their corresponding keys, $\{\mathsf{sk}_{f_i}\}_{i=1}^k$. An FE scheme achieves **adaptive simulation-based** (AD-SIM) security if there exists a PPT simulator

Sim such that, for any PPT adversary $\mathcal{A}$, the Real and the Ideal experiments are computationally indistinguishable.

We stress that we consider simulators that run in polynomial time. For the knowledgeable reader, it was shown by Boneh, Sahai and Waters [15] that AD-SIM-security is impossible to achieve for many challenge messages. While [15] provided the lower bound for the IBE functionality, the same argument easily extends to IPFE. Thus, as in [46], our security game must also be restricted to a single challenge ciphertext. Note that AD-SIM for a single ciphertext implies AD-IND for a single ciphertext, which in turn implies AD-IND for many ciphertexts [26]. Hence, AD-SIM for a single ciphertext is still the strongest definition of security for IPFE.

2.4 Hardness Assumptions

Our first scheme relies on the standard Decision Diffie-Hellman DDH assumption in ordinary (i.e., non-pairing-friendly) cyclic groups.

Definition 3. *In a cyclic group $\mathbb{G}$ of prime order p, the* **Decision Diffie-Hellman Problem** *(DDH) in $\mathbb{G}$, is to distinguish the distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), with $a, b, c \hookleftarrow \mathbb{Z}_p$. The* **Decision Diffie-Hellman** *assumption is the intractability of DDH for any PPT algorithm $\mathcal{D}$.*

Our second scheme relies on Paillier's composite residuosity assumption.

Definition 4 ([38]). *Let p, q be prime numbers and $N = pq$. The* **Decision Composite Residuosity** (DCR) *assumption states that the following two distributions are computationally indistinguishable:*

$$\left\{t_0^N \bmod N^2 \mid t_0 \leftarrow U(\mathbb{Z}_N^\star)\right\} \overset{c}{\approx} \left\{t \mid t \leftarrow U(\mathbb{Z}_{N^2}^\star)\right\}$$

Our third construction builds on the Learning-With-Errors (LWE) problem, which is known to be at least as hard as certain standard lattice problems in the worst case [40, 16].

Definition 5. *Let q, α, m be functions of a parameter n. For a secret $\mathbf{s} \in \mathbb{Z}_q^n$, the distribution $A_{q,\alpha,\mathbf{s}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is obtained by sampling $\mathbf{a} \hookleftarrow \mathbb{Z}_q^n$ and an $e \hookleftarrow D_{\mathbb{Z},\alpha q}$, and returning $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^{n+1}$. The* **Learning With Errors** *(LWE) problem $\mathsf{LWE}_{q,\alpha,m}$ is as follows: For $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$, the goal is to distinguish between the distributions:*

$$D_0(\mathbf{s}) := U(\mathbb{Z}_q^{m \times (n+1)}) \quad \textit{and} \quad D_1(\mathbf{s}) := (A_{q,\alpha,\mathbf{s}})^m.$$

We say that a PPT algorithm $\mathcal{A}$ solves $\mathsf{LWE}_{q,\alpha}$ if it distinguishes $D_0(\mathbf{s})$ and $D_1(\mathbf{s})$ with non-negligible advantage (over the random coins of $\mathcal{A}$ and the randomness of the samples), with non-negligible probability over the randomness of $\mathbf{s}$.

3 Adaptive Simulation-Based Security from DDH

In this section, we first recall the IPFE scheme of [8]. Abdalla *et al.* [3] previously showed that this construction provides simulation-based security for selective adversaries. In [46], Wee gave a proof of simulation-based security for semi-adaptive adversaries. We provide a proof that handles adaptive adversaries without any modification in the original scheme.

Setup$(1^\lambda, 1^\ell)$**:** Choose a cyclic group $\mathbb{G}$ of prime order $q > 2^\lambda$ with generators $g, h \hookleftarrow U(\mathbb{G})$. Then, for each $i \in \{1, \ldots, \ell\}$, sample $s_i, t_i \hookleftarrow U(\mathbb{Z}_q)$ and compute $h_i = g^{s_i} \cdot h^{t_i}$. Define $\mathsf{msk} := \{s_i, t_i\}_{i=1}^{\ell}$ and

$$\mathsf{mpk} := \Big(\mathbb{G}, g, h, \{h_i\}_{i=1}^{\ell}\Big).$$

Keygen$(\mathsf{msk}, \boldsymbol{y})$**:** To generate a key for the vector $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}_q^\ell$, compute $\mathsf{sk}_{\boldsymbol{y}} = (s_{\boldsymbol{y}}, t_{\boldsymbol{y}}) = (\sum_{i=1}^{\ell} s_i \cdot y_i, \sum_{i=1}^{\ell} t_i \cdot y_i) = (\langle \boldsymbol{s}, \boldsymbol{y} \rangle, \langle \boldsymbol{t}, \boldsymbol{y} \rangle)$.

Encrypt$(\mathsf{mpk}, \boldsymbol{x})$**:** To encrypt a vector $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$, sample $r \hookleftarrow \mathbb{Z}_q$ and compute

$$C = g^r, \qquad D = h^r, \qquad \{E_i = g^{x_i} \cdot h_i^r\}_{i=1}^{\ell}.$$

Return $C_{\boldsymbol{x}} = (C, D, E_1, \ldots, E_\ell)$.

Decrypt$(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, C_{\boldsymbol{x}})$**:** Given $\mathsf{sk}_{\boldsymbol{y}} = (s_{\boldsymbol{y}}, t_{\boldsymbol{y}})$, compute

$$E_{\boldsymbol{y}} = \Big(\prod_{i=1}^{\ell} E_i^{y_i}\Big) / (C^{s_{\boldsymbol{y}}} \cdot D^{t_{\boldsymbol{y}}}).$$

Then, compute and output $\log_g(E_{\boldsymbol{y}})$.

Correctness. Note that $\prod_{i=1}^{\ell} E_i^{y_i} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g^{r\langle \boldsymbol{s}, \boldsymbol{y} \rangle} \cdot h^{r\langle \boldsymbol{t}, \boldsymbol{y} \rangle} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot C^{s_{\boldsymbol{y}}} \cdot D^{t_{\boldsymbol{y}}}$, which implies $E_{\boldsymbol{y}} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$. The decryption algorithm can thus recover $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ mod q by solving a discrete logarithm instance in a small interval, by restricting messages and keys so as to have $|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq L$, for some polynomially bounded $L = \mathsf{poly}(\lambda)$. In this case, the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ can be recovered in $\tilde{O}(L^{1/2})$ time using [39].

Theorem 1. *The scheme provides simulation-based security against adaptive adversaries under the* DDH *assumption.*

Proof. To prove the result, we first describe a PPT simulator before showing that, under the DDH assumption, the adversary cannot distinguish the ideal experiment from the real experiment.

In both experiments, we know that the adversary $\mathcal{A}$ can obtain private keys for up to $\ell - 1$ linearly independent vectors. We assume w.l.o.g. that $\mathcal{A}$ makes private keys queries for exactly $\ell - 1 = \ell_0 + \ell_1$ independent vectors, which we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell-1} \in \mathbb{Z}_q^\ell$. Among these vectors, we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell_0}$ the vectors queried by $\mathcal{A}$ *before* the challenge phase while $\boldsymbol{y}_{\ell_0+1}, \ldots, \boldsymbol{y}_{\ell_0+\ell_1}$ stand for the post-challenge private key queries. In the challenge phase, we denote by $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_\ell^\star) \in \mathbb{Z}_q^\ell$ the message chosen by $\mathcal{A}$. The simulator $(\mathsf{Setup}^\star, \mathsf{Keygen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{Keygen}_1^\star)$ proceeds in the following way.

$\mathsf{Setup}^\star(1^\lambda, 1^\ell)$: This algorithm is identical to Setup except that $\omega = \log_g(h)$ is included in the master secret key. It outputs

$$\mathsf{mpk}^\star := \Big(\mathbb{G}, g, h, \{h_i\}_{i=1}^{\ell}\Big).$$

and $\mathsf{msk}^\star = (\omega, \boldsymbol{s}, \boldsymbol{t})$.

$\mathsf{Keygen}_0^\star(\mathsf{msk}^\star, \boldsymbol{y})$: This algorithm is used to answer private key queries before the challenge phase and proceeds exactly like Keygen in the real scheme.

$\mathsf{Encrypt}^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \mathcal{V}, \{1^{|x_i^\star|}\}_{i=1}^{\ell})$: This algorithm takes as input $\mathsf{mpk}^\star, \mathsf{msk}^\star$, the lengths $\{1^{|x_i^\star|}\}_{i=1}^{\ell}$ of all coordinates of $\boldsymbol{x}^\star$ and a set

$$\mathcal{V} = \Big\{\{\boldsymbol{y}_j, z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j\rangle, \mathsf{sk}_{\boldsymbol{y}_j}\}_{j=1}^{\ell_0}\Big\}$$

containing all pre-challenge independent queries $\{\boldsymbol{y}_j\}_{j=1}^{\ell_0}$, the returned keys and the corresponding linear function evaluations $\{z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j\rangle\}_{j=1}^{\ell_0}$ for the challenge message $\boldsymbol{x}^\star$. The challenge ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ is simulated as follows.

1. Letting $\boldsymbol{z}_{\mathsf{pre}} = (z_1, \ldots, z_{\ell_0})^\top \in \mathbb{Z}_q^{\ell_0}$, compute an arbitrary $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^{\ell}$ such that $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \bmod q$, where

$$\mathbf{Y}_{\mathsf{pre}} = \begin{bmatrix} \boldsymbol{y}_1^\top \\ \vdots \\ \boldsymbol{y}_{\ell_0}^\top \end{bmatrix} \in \mathbb{Z}_q^{\ell_0 \times \ell}.$$

   Note that $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_\ell)^\top$ does not have to be small and can be obtained via Gaussian elimination.
2. Compute the ciphertext by sampling $r, r' \hookleftarrow U(\mathbb{Z}_q)$ uniformly and computing $(C^\star, D^\star) = (g^r, h^{r'})$ as well as

$$E_i^\star = g^{\bar{x}_i} \cdot C^{\star s_i} \cdot D^{\star t_i} \qquad \forall i \in [\ell].$$

Output the simulated ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ together with the state information $\mathsf{st}' = (\bar{\boldsymbol{x}}, r, r')$.

$\mathsf{Keygen}_1^\star(\mathsf{msk}^\star, \boldsymbol{y}, z = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle, \mathsf{st}')$: On input of $\mathsf{msk}^\star = (\omega, \boldsymbol{s}, \boldsymbol{t})$, a post-challenge query $\boldsymbol{y} \in \mathbb{Z}_q^{\ell}$, the evaluation $z = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle$ of the linear function $f_{\boldsymbol{y}}(\boldsymbol{x}^\star)$ on the message $\boldsymbol{x}^\star$ and the state information $\mathsf{st}' = (\bar{\boldsymbol{x}}, r, r') \in \mathbb{Z}_q^{\ell} \times \mathbb{Z}_q^2$, this algorithm computes

$$t'_{\boldsymbol{y}} = \langle \boldsymbol{t}, \boldsymbol{y}\rangle + \frac{1}{\omega \cdot (r' - r)} \cdot (\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - z) \mod q. \tag{3.1}$$

$$s'_{\boldsymbol{y}} = \langle \boldsymbol{s}, \boldsymbol{y}\rangle - \frac{1}{(r' - r)} \cdot (\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - z) \mod q.$$

and returns $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$.

Observe that the ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ produced by $\mathsf{Encrypt}^\star$ is distributed in such a way that $(C^\star, D^\star) = (g^r, g^{\omega\cdot(r+(r'-r))})$ and

$$(E_1^\star, \ldots, E_\ell^\star) = g^{\bar{\boldsymbol{x}}+\omega\cdot(r'-r)\cdot\boldsymbol{t}} \cdot (h_1, \ldots, h_\ell)^r,$$

so that, for any $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in \mathbb{Z}_q^\ell$, we have

$$\prod_{i=1}^{\ell} E_i^{\star y_i} = g^{\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle + \omega\cdot(r'-r)\cdot\langle \boldsymbol{t}, \boldsymbol{y}\rangle} \cdot (g^{\langle \boldsymbol{s}, \boldsymbol{y}\rangle} \cdot h^{\langle \boldsymbol{t}, \boldsymbol{y}\rangle})^r,$$

which implies

$$\prod_{i=1}^{\ell} E_i^{\star y_i} / (C^{\star s'_{\boldsymbol{y}}} \cdot D^{\star t'_{\boldsymbol{y}}}) = g^z.$$

This shows that decrypting the simulated ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ using the simulated key $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$ yields $z = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle$, as required.

We now proceed to show that the simulation is computationally indistinguishable from the real experiment under the DDH assumption.

The proof uses a sequence of games that begins with a game in which the challenger interacts with the adversary as in real experiment and ends with a game where the challenger interacts with the adversary as in the ideal experiment. For Game_i and Game_j we denote by $\mathsf{Adv}_{ij}(\mathcal{A})$ the advantage of a PPT algorithm $\mathcal{A}$ in distinguishing between Game_i and Game_j. Formally the challenger $\mathcal{C}$ flips a coin $b \hookleftarrow \{0,1\}$. If $b = 0$ it interacts with the adversary as in Game_i, else it interacts as in Game_j. At the end of the interaction $\mathcal{A}$ will have to make its guess $b' \in \{0,1\}$. We define $\mathsf{Adv}_{ij}(\mathcal{A}) := \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Game_0: In this game the challenger interacts with the adversary as in the real experiment.

Game_1: We modify the generation of the ciphertext $C_{\boldsymbol{x}}^\star = (C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$. Namely, the experiment $\mathcal{B}$ first computes

$$C^\star = g^r \quad \text{and} \quad D^\star = h^r, \tag{3.2}$$

for a randomly sampled $r \hookleftarrow \mathbb{Z}_q$. Then, it uses $\mathsf{msk} := \{s_i, t_i\}_{i=1}^\ell$ to compute

$$E_i^\star = g^{x_i^\star} \cdot C^{\star s_i} \cdot D^{\star t_i}. \tag{3.3}$$

It can be observed that $C_{\boldsymbol{x}}^\star = (C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ has the same distribution as in Game 0. We hence have $\mathsf{Adv}_{01}(\mathcal{A}) = 0$.

Game_2: We modify again the generation of $C_{\boldsymbol{x}}^\star = (C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$. Namely, instead of computing the pair $(C^\star, D^\star)$ as in (3.2), the experiment samples $r, r' \hookleftarrow U(\mathbb{Z}_q)$ and sets

$$C^\star = g^r \quad \text{and} \quad D^\star = h^{r'}.$$

The ciphertext components $(E_1^\star, \ldots, E_\ell^\star)$ are still computed as per (3.3). Under the DDH assumption, this modification should not significantly affect $\mathcal{A}$'s view and we have $\mathsf{Adv}_{12}(\mathcal{A}) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(1^\lambda)$.

Game_3: In this game, the challenger runs exactly the ideal experiment with the adversary. Lemma 5 shows that $\mathsf{Adv}_{23}(\mathcal{A}) = 0$.

Combining the above, we find

$$|\Pr[1 \leftarrow \mathbf{Exp}_{\mathcal{A}}^{\mathsf{Real}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Exp}_{\mathcal{A}}^{\mathsf{Ideal}}(1^\lambda)]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(1^\lambda),$$

as claimed. □

Lemma 5. *The advantage of an adversary $\mathcal{A}$ in distinguishing between Game_2 and Game_3 is 0.*

Proof. To prove the result, we define the following two variants of these games.

Game_2': This game is identical to Game_2 except that, at the outset of the game, the challenger chooses a random vector $\boldsymbol{\Delta x} \hookleftarrow U(\mathbb{Z}_q^\ell)$. It interacts with $\mathcal{A}$ as in Game_2 until the challenge phase, at which point it samples an arbitrary vector $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^\ell$ satifying $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \mathbf{Y}_{\mathsf{pre}} \cdot \boldsymbol{x}^\star \bmod q$, where $\mathbf{Y}_{\mathsf{pre}} \in \mathbb{Z}_q^{\ell_0 \times \ell}$ is the matrix whose rows are the first ℓ_0 independent key queries. At this point, the challenger checks whether $\boldsymbol{\Delta x} = \bar{\boldsymbol{x}} - \boldsymbol{x}^\star \bmod q$ (we call Guess this event). If not, it aborts the interaction with $\mathcal{A}$ and replaces $\mathcal{A}$'s output with 0. Otherwise, it proceeds like Game_2 and outputs whatever $\mathcal{A}$ outputs. Since $\boldsymbol{\Delta x}$ is drawn uniformly and independently of $\mathcal{A}$'s view, we have $\Pr[\mathsf{Guess}] = 1/q^\ell$.

Game_3': This game is like Game_3, except that, at the very beginning of the game, the challenger chooses a random $\boldsymbol{\Delta x} \hookleftarrow U(\mathbb{Z}_q^\ell)$. It proceeds like Game_3 until the challenge phase, at which point it samples an arbitrary $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^\ell$ satifying $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \bmod q$. Then, it checks whether $\boldsymbol{\Delta x} = \bar{\boldsymbol{x}} - \boldsymbol{x}^\star \bmod q$ (we call Guess this event). If not, it aborts and replaces $\mathcal{A}$'s output with 0. Otherwise, it proceeds identically to Game_3 and outputs the same result as $\mathcal{A}$.

Now, we claim that Game_2' and Game_3' are identical. To see this, we first note that, conditionally on $\neg\mathsf{Guess}$, both games output 0. If Guess occurs, we observe that Game_3' is identical to Game_2' when the master secret key is replaced by $(\boldsymbol{s}', \boldsymbol{t}') \in \mathbb{Z}_q^\ell \times \mathbb{Z}_q^\ell$, where

$$\begin{aligned}
t_i' &= t_i + \frac{1}{\omega \cdot (r' - r)} \cdot \boldsymbol{\Delta x} \\
&= t_i + \frac{1}{\omega \cdot (r' - r)} \cdot (\bar{x}_i - x_i^\star) \bmod q \qquad\qquad \forall i \in [\ell] \\
s_i' &= s_i - \frac{1}{r' - r} \cdot \boldsymbol{\Delta x} \\
&= s_i - \frac{1}{r' - r} \cdot (\bar{x}_i - x_i^\star) \bmod q.
\end{aligned}$$

Indeed, $(\boldsymbol{s}', \boldsymbol{t}')$ has the same distribution as $(\boldsymbol{s}, \boldsymbol{t})$ conditionally on mpk. By construction, we also have $\langle \boldsymbol{s}', \boldsymbol{y} \rangle = \langle \boldsymbol{s}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{t}', \boldsymbol{y} \rangle = \langle \boldsymbol{t}, \boldsymbol{y} \rangle$ in all pre-challenge queries $\boldsymbol{y} \in \mathbb{Z}_q^\ell$. Moreover, we have

$$g^{\bar{\boldsymbol{x}} + \omega \cdot (r'-r) \cdot \boldsymbol{t}} \cdot (h_1, \ldots, h_\ell)^r = g^{\boldsymbol{x}^\star + \omega \cdot (r'-r) \cdot \boldsymbol{t}'} \cdot (h_1, \ldots, h_\ell)^r.$$

Finally, answering post-challenge queries $\boldsymbol{y} \in \mathbb{Z}_q^\ell$ using $(\boldsymbol{s}', \boldsymbol{t}')$ gives exactly the distribution (3.1). This implies that the games are indeed identical, therefore $\mathsf{Adv}'_{23} = 0$.

To conclude, notice that any adversary $\mathcal{A}$ that can distinguish between Game_2 and Game_3 can be used to distinguish between Game'_2 and Game'_3, with a loss factor of q^ℓ in the advantage:

$$\mathsf{Adv}'_{23} = \frac{1}{q^\ell} \cdot \mathsf{Adv}_{23}(\mathcal{A})$$

This holds since the probability that $\mathcal{A}$ outputs the correct bit b' when distinguishing between Game'_2 and Game'_3 is equal to:

$$\Pr[b' = b] = \Pr[b' = b | \mathsf{Guess}] \cdot \Pr[\mathsf{Guess}] + \Pr[b' = b | \overline{\mathsf{Guess}}] \cdot \Pr[\overline{\mathsf{Guess}}]$$

which is equivalent to:

$$\Pr[b' = b] - \frac{1}{2} = \left(\Pr[b' = b | \mathsf{Guess}] - \frac{1}{2} \right) \cdot \Pr[\mathsf{Guess}]$$

By considering the equality in absolute value, we get the desired relation between the advantages. □

While efficient and based on a standard assumption, the scheme of [8] is restricted to the evaluation of inner products confined in a small interval. In the next section, we show that our proof can be adapted to the Paillier-based constructions of [8, 11], which make it possible to evaluate inner products over exponentially large intervals.

4 Adaptive Simulation-Based Security for Inner Products over $\mathbb{Z}$ from DCR

This section shows that a variant of the Paillier-based IPFE scheme of Agrawal *et al.* [8] can also be proved simulation-secure for adaptive adversaries. Like the first DCR-based construction of [8], it evaluates inner products over the integers. Our variant differs from [8] in that master secret keys are no longer sampled from a Gaussian distribution but are rather sampled uniformly in a large interval.

In [11], Benhamouda *et al.* also considered secret keys sampled from a uniform distribution over an interval. Their motivation was to obtain indistinguishability-based security under chosen-ciphertext attacks for adaptive adversaries. Our goal differs from theirs in that we do not consider chosen-ciphertext attacks but rather

focus on achieving simulation-based security. To this end, we have to sample master secret keys from a significantly larger interval.

The reason why we need larger master secret keys is that, in the challenge phase, our simulator has to sample a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ that should satisfy an equation of the form $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^k$, for some given $\mathbf{Y}_{\mathsf{pre}} \in \mathbb{Z}^{k \times \ell}$ and $\boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^k$, in order to be consistent with responses $\boldsymbol{z}_{\mathsf{pre}} = (z_1, \ldots, z_k)$ to all pre-challenge queries. For lack of a short basis for the lattice $\mathbf{Y}_{\mathsf{pre}}^{\perp} := \{\boldsymbol{x} \in \mathbb{Z}^\ell : \mathbf{Y}_{\mathsf{pre}} \cdot \boldsymbol{x} = \mathbf{0}\}$, our simulator can only sample a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ with large entries. At each post-challenge query $\boldsymbol{y} \in \mathbb{Z}^\ell$, the simulator has to "program" the returned functional secret key in such a way that it decrypts the simulated ciphertext to the value $z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$ dictated by the oracle. For this purpose, the "programmed" key $\mathsf{sk}'_{\boldsymbol{y}}$ must consist of the sum (over $\mathbb{Z}$) of the real key $\mathsf{sk}_{\boldsymbol{y}} = \langle \boldsymbol{s}, \boldsymbol{y} \rangle$ and a multiple of the difference $z - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle$ between the function evaluation $f_{\boldsymbol{y}}(\bar{\boldsymbol{x}}) = \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle$ and the oracle value $z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$. Since $z - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle$ may be large over $\mathbb{Z}$, we need to sample the entries of $\boldsymbol{s} \in \mathbb{Z}^\ell$ from a sufficiently wide interval so as to "drown" the statistical discrepancy between the distributions of the master secret $\boldsymbol{s} \in \mathbb{Z}^\ell$ and its shifted variant $\boldsymbol{s}' = \boldsymbol{s} + \gamma \cdot (\boldsymbol{x}^\star - \bar{\boldsymbol{x}}) \in \mathbb{Z}^\ell$ for which $\mathsf{sk}'_{\boldsymbol{y}} = \langle \boldsymbol{s}', \boldsymbol{y} \rangle$. Since RSA moduli should asymptotically contain $\lambda^3/\mathrm{polylog}(\lambda)$ bits to resist factorization attacks, we need to sample each entry of $\boldsymbol{s} \in \mathbb{Z}^\ell$ from an interval of cardinality $O(2^{\ell^2 \cdot \lambda^3/\mathrm{polylog}(\lambda)})$. Despite somewhat large secret keys, the scheme remains computationally efficient as only one exponentiation with a large exponent $\mathsf{sk}_{\boldsymbol{y}}$ suffices to decrypt. We see it as an interesting open problem to obtain shorter keys while retaining simulation-based security.

Setup$(1^\lambda, 1^\ell, X, Y)$: Choose safe primes $p = 2p' + 1$ and $q = 2q' + 1$ with p', q' also primes, such that $\ell XY < N/2$, where $N = pq$. Sample $g' \leftarrow U(\mathbb{Z}^*_{N^2})$ and set $g := g'^{2N} \bmod N^2$. Next for each $i \in [\ell]$ sample $s_i \leftarrow U([-S, S] \cap \mathbb{Z})$, where $S = 2^{\lambda+\ell-1} \cdot \bar{X}^{\ell-1} \cdot \ell N^2$ and $\bar{X} := X + \ell \cdot (\sqrt{\ell} Y)^\ell$ and then compute $h_i = g^{s_i} \bmod N^2$. Define $\mathsf{msk} := \boldsymbol{s} = (s_1, \ldots, s_\ell)^\top \in \mathbb{Z}^\ell$ and $\mathsf{mpk} := \big(N, g, \{h_i\}_{i=1}^\ell, X, Y\big)$

Keygen$(\mathsf{msk}, \boldsymbol{y})$: To generate a secret key from the vector $\boldsymbol{y} \in [-Y, Y]^\ell$ using $\mathsf{msk} = \boldsymbol{s} = (s_1, \ldots, s_\ell)^\top$, compute $\mathsf{sk}_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle = \sum_{i=1}^\ell s_i \cdot y_i \in \mathbb{Z}$.

Encrypt$(\mathsf{mpk}, \boldsymbol{x})$: Given the public key mpk, to encrypt a message $\boldsymbol{x} \in [-X, X]^\ell$, sample $r \leftarrow U(\{0, 1, \ldots, N/4\})$ and compute

$$c_0 = g^r \bmod N^2, \qquad c_i = (1 + x_i N) \cdot h_i^r \bmod N^2 \qquad \forall i \in [\ell]$$

and output $\mathbf{c} = (c_0, \{c_i\}_{i=1}^\ell) \in \big(\mathbb{Z}^*_{N^2}\big)^{\ell+1}$.

Decrypt$(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, \mathbf{c})$: On input of a functional decryption key $\mathsf{sk}_{\boldsymbol{y}}$ and a ciphertext $\mathbf{c} = (c_0, c_1, \ldots, c_\ell)$, compute

$$\mathbf{c}_{\boldsymbol{y}} = c_0^{-\mathsf{sk}_{\boldsymbol{y}}} \cdot \prod_{i=1}^{\ell} c_i^{y_i} \bmod N^2$$

Then output $\log_{1+N}(\mathbf{c}_{\boldsymbol{y}}) = \frac{\mathbf{c}_{\boldsymbol{y}} - 1 \bmod N^2}{N}$.

Correctness: Suppose that we want to decrypt $\boldsymbol{c} = \{c_i\}_{i=0}^{\ell}$ using $\mathsf{sk}_{\boldsymbol{y}} = \langle \boldsymbol{s}, \boldsymbol{y} \rangle$. Observe that we have the following equalities modulo N^2:

$$\prod_{i=1}^{\ell} c_i^{y_i} = \prod_{i=1}^{\ell} (1 + x_i N)^{y_i} \cdot g^{r \cdot s_i y_i} = (1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g^{r \cdot \langle \boldsymbol{s}, \boldsymbol{y} \rangle} = (1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot c_0^{\langle \boldsymbol{s}, \boldsymbol{y} \rangle},$$

so that $\boldsymbol{c}_{\boldsymbol{y}} = (1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \bmod N^2$. Recall that $(1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} = 1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle \cdot N \bmod N^2$, so that computing discrete logarithms in the subgroup generated by $1+N$ is easy. This enables the computation of $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \bmod N$. By the choice of parameters we have $|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \ell \cdot \|\boldsymbol{x}\|_\infty \|\boldsymbol{y}\|_\infty \leq \ell \cdot X \cdot Y < N/2$, so we actually recover $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ computed over $\mathbb{Z}$.

Theorem 2. *Under the* DCR *assumption, the above construction achieves adaptive simulation-based security.*

Proof. To prove the theorem we first describe the PPT simulator and show that under the DCR assumption the real experiment is indistinguishable from the ideal experiment. The simulator proceeds as follows.

Setup$^\star(1^\lambda, 1^\ell, X, Y)$: This algorithm chooses safe primes $p = 2p' + 1$ and $q = 2q' + 1$ such that $\ell XY < N/2$, and sets $N = pq$. It samples $g' \leftarrow U(\mathbb{Z}_{N^2}^*)$ and sets $g := g'^{2N} \bmod N^2$. Next, for each $i \in [\ell]$, it samples $s_i \leftarrow U([-S, S] \cap \mathbb{Z})$, where $S = 2^{\lambda+\ell-1} \cdot \bar{X}^{\ell-1} \cdot \ell N^2$ and $\bar{X} := X + \ell \cdot (\sqrt{\ell} Y)^\ell$, and computes $h_i = g^{s_i} \bmod N^2$. It defines the master secret key $\mathsf{msk}^\star = (\boldsymbol{s}, p, q)$, where $\boldsymbol{s} = (s_1, \ldots, s_\ell)^\top$, and the master public key $\mathsf{mpk}^\star = \left(N, g, \{h_i\}_{i=1}^{\ell}, X, Y\right)$

Keygen$_0^\star(\mathsf{msk}^\star, \boldsymbol{y})$: This algorithm is used to generate all the pre-challenge functional decryption queries. To generate a secret key for $\boldsymbol{y} \in [-Y, Y]^\ell$, it computes and outputs $\mathsf{sk}_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle = \sum_{i=1}^{\ell} s_i \cdot y_i \in \mathbb{Z}$.

Encrypt$^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \{(\boldsymbol{y}_1, z_1), (\boldsymbol{y}_2, z_2), \ldots, (\boldsymbol{y}_k, z_k)\})$: Given $\mathsf{mpk}^\star$, $\mathsf{msk}^\star$ and all the pre-challenge pairs $(\boldsymbol{y}_j, z_j) \in [-Y, Y]^\ell \times \mathbb{Z}$, where $z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j \rangle \in \mathbb{Z}$ and $\boldsymbol{x}^\star$ is the challenge message, it first computes a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ such that $\langle \bar{\boldsymbol{x}}, \boldsymbol{y}_j \rangle = z_j$ for all $j \in [k]$ by applying Corollary 1. Note that $\|\bar{\boldsymbol{x}}\|_\infty \leq (\ell - k) \cdot (\sqrt{k} Y)^k \leq \ell \cdot (\sqrt{\ell} Y)^\ell$. Next, it samples $a \leftarrow U(\mathbb{Z}_N^*)$ and $b \leftarrow U(\mathbb{Z}_{N'})$, where $N' = p'q'$, and computes

$$c_0^\star = (1 + aN) \cdot g^b \bmod N^2, \qquad c_i^\star = (1 + \bar{x}_i N) \cdot (c_0^\star)^{s_i} \bmod N^2 \qquad \forall i \in [\ell].$$

It outputs the simulated ciphertext $\mathbf{c}^\star = \left(c_0^\star, \{c_i^\star\}_{i=1}^{\ell}\right) \in \left(\mathbb{Z}_{N^2}^*\right)^{\ell+1}$ together with the state information $\mathsf{st} := (\bar{\boldsymbol{x}}, a, N')$

Keygen$_1^\star(\mathsf{msk}^\star, (\boldsymbol{y}, z = \langle \boldsymbol{y}, \boldsymbol{x}^\star \rangle), \mathsf{st})$: This algorithm handles post-challenge key queries as follows. Upon receiving a pair $(\boldsymbol{y}, z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle)$, it first computes $u, v \in \mathbb{Z}$ such that $uN + vN' = 1$ and $\gamma := (a^{-1} \bmod N) \cdot vN' \bmod NN'$ then computes and outputs

$$\mathsf{sk}'_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle - \gamma \cdot (z - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle) \in \mathbb{Z}.$$

In order to prove that the real experiement is computationally indistinguishable from the ideal experiment, we use a sequence of games. We denote by $\mathsf{Adv}_{ij}(\mathcal{A})$ the advantage of an adversary $\mathcal{A}$ in distinguishing between Game_i and Game_j. More precisely, a challenger $\mathcal{C}$ flips a coin $b \leftarrow \{0,1\}$. If $b = 0$ the challenger interacts with the adversary $\mathcal{A}$ as in Game_i while, if $b = 1$, it interacts as in Game_j. At the end of the interaction, $\mathcal{A}$ outputs $b' \in \{0,1\}$. The advantage is defined as $\mathsf{Adv}_{ij}(\mathcal{A}) := \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Game_0 : This is the real game in which the challenger generates the parameters and interacts with the adversary as in the real experiment.

Game_1 : This game is exactly as the previous one except that the challenge ciphertext is computed as follows: $r \leftarrow U(\{0,1,\ldots,N/4\})$ is sampled and

$$c_0^\star = g^r \bmod N^2, \qquad c_i^\star = (1 + x_i^\star N) \cdot (c_0^\star)^{s_i} \bmod N^2, \text{ for } i \in [\ell]$$

This is possible since the challenger knows the secret key $\mathsf{msk} = \left(\{s_i\}_{i=1}^{\ell}\right)$. Notice that Game_0 is identical to Game_1. So, $\mathsf{Adv}_{01}(\mathcal{A}) = 0$.

Game_2 : In this game, we modify the computation of $c_0^\star$. In the challenge phase, the challenger samples $r \leftarrow U(\mathbb{Z}_{N'})$, where $N' = p'q'$, and computes $c_0^\star := g^r \bmod N^2$. By Lemma 1, the statistical distance between $U(\{0,1,2,\ldots,N/4\}) \bmod N'$ and $U(\mathbb{Z}_{N'})$ is $< \frac{1}{p} + \frac{1}{q}$, which is negligible. Hence, Game_1 and Game_2 are statistically indistinguishable. More precisely, we have $\mathsf{Adv}_{12}(\mathcal{A}) < 1/p + 1/q$.

Game_3 : The game is like Game_2, except that $c_0^\star$ is generated by sampling $t \hookleftarrow U(\mathbb{Z}_{N^2}^*)$ and computing $c_0^\star := t^2 \bmod N^2$. Under the DCR assumption, Game_2 and Game_3 are computationally indistinguishable. Indeed, in Game_2, as long as g has order N', the distribution $\{g^r \mid r \hookleftarrow U(\mathbb{Z}_{N'})\}$ is the uniform distribution in the subgroup of $2N$-th residues. The DCR assumption implies that the latter distribution computationally indistinguishable from the distribution $\{t^2 \bmod N^2 \mid t \hookleftarrow U(\mathbb{Z}_{N^2}^*)\}$. Since a random $2N$-th residue g generates the entire subgroup of $2N$-th residues with probability $\frac{\varphi(N')}{N'} = 1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{N'}$, we obtain

$$\left(1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{N'}\right) \cdot \mathsf{Adv}_{23}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{DCR}}(\mathcal{B}).$$

Game_4 : In this game, we sample $a \leftarrow U(\mathbb{Z}_N^*)$ and $b \leftarrow U(\mathbb{Z}_{N'})$ and compute $c_0^\star := (1 + aN) \cdot g^b \bmod N^2$. Observe that $\{t^2 \bmod N^2 \mid t \hookleftarrow U(\mathbb{Z}_{N^2}^*)\}$ is the same as the distribution $\{(1+\alpha N) \cdot g^\beta \bmod N^2 \mid \alpha \hookleftarrow U(\mathbb{Z}_N),\ \beta \hookleftarrow U(\mathbb{Z}_{N'})\}$. Therefore the statistical distance between the view of the adversary in Game_3 and Game_4 is bounded by $\Delta(a,\alpha) < \frac{1}{p} + \frac{1}{q}$. So, these games are statistically indistinguishable and $\mathsf{Adv}_{34}(\mathcal{A}) < 1/p + 1/q$.

Game_5 : This is the ideal experiment where the adversary interacts with the simulator. Lemma 6 shows that Game_5 and Game_4 are statistically indistinguishable, which yields the stated result.

Putting the above altogether, we obtain that a PPT adversary $\mathcal{A}$ that can distinguish between the real and the ideal experiment implies an efficient DCR distinguisher $\mathcal{B}$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Real\text{-}Ideal}}(\mathcal{A}) &= |\Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{Real}}_{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{Ideal}}_{\mathcal{A}}(1^\lambda)]| \\ &\leq \frac{N'}{\varphi(N')} \cdot \mathsf{Adv}^{\mathsf{DCR}}_{\mathcal{B}}(1^\lambda) + \frac{2}{p} + \frac{2}{q} + 2^{-\lambda}.\end{aligned}$$

□

Lemma 6. *The advantage of any distinguisher between* Game_4 *and* Game_5 *is statistically negligible and* $\mathsf{Adv}_{45}(\mathcal{A}) \leq 2^{-\lambda}$.

Proof. In order to prove the claim, we simultaneously define Game'_4 and Game'_5 as follows. For each $k \in \{4, 5\}$, define Game'_k identically to Game_k except that, at the outset of the game, the challenger samples $\Delta\boldsymbol{x} \hookleftarrow U([-\bar{X}, \bar{X}]^\ell)$, where $\bar{X} = X + \ell \cdot (\sqrt{\ell}Y)^\ell$. Before generating the challenge ciphertext, the challenger uses Corollary 1 to compute $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ such that $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \mathbf{Y}_{\mathsf{pre}} \cdot \boldsymbol{x}^\star$, where $\mathbf{Y}_{pre}$ is the matrix obtained by stacking up the (linearly independent) transposed vectors $\boldsymbol{y}^\top$ occurring in pre-challenge queries. If $\Delta\boldsymbol{x} = \boldsymbol{x}^\star - \bar{\boldsymbol{x}}$ (we call this event Guess), the challenger proceeds as in Game_k. Otherwise, the challenger aborts the game and replaces $\mathcal{A}$'s output b' by a random bit. We claim that any adversary $\mathcal{A}$ that can distinguish between Game_4 and Game_5 with advantage $\mathsf{Adv}_{45}(\mathcal{A})$ can be used to distinguish between Game'_4 and Game'_5 with advantage

$$\mathsf{Adv}'_{45}(\mathcal{A}) = \frac{1}{(2\bar{X})^\ell} \cdot \mathsf{Adv}_{45}(\mathcal{A}). \tag{4.1}$$

Indeed, the probability that $\mathcal{A}$ outputs the correct bit b' when distinguishing between Game'_4 and Game'_5 is equal to

$$\Pr[b' = b] = \Pr[b' = b|\mathsf{Guess}] \cdot \Pr[\mathsf{Guess}] + \Pr[b' = b|\overline{\mathsf{Guess}}] \cdot \Pr[\overline{\mathsf{Guess}}]$$

which is equivalent to

$$\Pr[b' = b] - \frac{1}{2} = \left(\Pr[b' = b|\mathsf{Guess}] - \frac{1}{2}\right) \cdot \Pr[\mathsf{Guess}]$$

By considering the equality in absolute value, we obtain (4.1).

Next, we claim that $\mathsf{Adv}'_{45}(\mathcal{A}) \leq (2\bar{X})^{-\ell} \cdot 2^{-\lambda}$, which implies that Game_4 and Game_5 are indistinguishable. To see this, observe that, when Guess occurs, Game'_5 is identical to a modification of Game'_4 where the master secret key has been replaced by

$$s'_i = s_i - \gamma \cdot \Delta\boldsymbol{x}_i \in \mathbb{Z}, \qquad \forall i \in [\ell]$$

where $\gamma = (a^{-1} \bmod N) \cdot vN' \bmod NN'$ is determined by the Bézout coefficient v for which $uN + vN' = 1$ (and thus $vN' = 1 \bmod N$) and the element $a \in \mathbb{Z}^*_N$ which used to compute $c^\star_0 = (1 + aN) \cdot g^b \bmod N^2$ in the challenge ciphertext.

(Note that a and v can be chosen by the challenger at the beginning of the game, so that we can define a game where the challenger uses $\{s'_i\}_i$ instead of $\{s_i\}_i$). With this new master secret key $\boldsymbol{s}' = (s'_1, \ldots, s'_\ell)$, we have $g^{s_i} = g^{s'_i} \bmod N^2$ for all $i \in [\ell]$ and $\langle \boldsymbol{s}, \boldsymbol{y} \rangle = \langle \boldsymbol{s}', \boldsymbol{y} \rangle$ for all pre-challenge queries $\boldsymbol{y} \in \mathbb{Z}^\ell$. We thus obtain

$$\mathsf{Adv}'_{45}(\mathcal{A}) \leq \Delta(\boldsymbol{s}', \boldsymbol{s}) \leq (2\bar{X})^{-\ell} \cdot 2^{-\lambda},$$

where the last inequality follows from the fact that

$$\Delta(\boldsymbol{s}', \boldsymbol{s}) \leq \sum_{i=1}^{\ell} \Delta(s'_i, s_i) \overset{\text{Lemma 2}}{\leq} \ell \cdot \frac{\|\gamma \cdot \Delta \boldsymbol{x}\|_\infty}{2S} \leq \frac{NN' \cdot \bar{X}}{2^{\lambda+\ell} \cdot \bar{X}^{\ell-1} \cdot N^2} \leq (2\bar{X})^{-\ell} \cdot 2^{-\lambda}.$$

□

The above DCR-based construction is stateless and evaluates inner products over $\mathbb{Z}$. In Section 5, we describe a generic construction of simulation-secure IPFE with stateful key generation, which allows evaluating inner products modulo a prime or a composite. This generic construction can be instantiated under the DCR and LWE assumptions.

5 Adaptive Simulation-Based Security for Inner Products mod p from LWE

In this section we construct an adaptively simulation secure FE scheme (AdSimIPFE) for inner products modulo some prime p. In more detail, the messages and keys are chosen from $\mathbb{Z}_p^\ell$ and the inner product is computed over $\mathbb{Z}_p$.

We denote our scheme by $\mathsf{AdSimIPFE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. Our construction is based on the scheme of Agrawal *et al.* [8] for inner products modulo a prime p satisfying adaptive indistinguishability from LWE. We denote this scheme by $\mathsf{IPFE} = (\mathsf{IPFE.Setup}, \mathsf{IPFE.Keygen}, \mathsf{IPFE.Encrypt}, \mathsf{IPFE.Decrypt})$, and require it to support messages and keys of length $L = 2\ell$.

Our construction is generic except that it requires the underlying scheme IPFE to satisfy the property that functional keys for vectors that are linearly dependent on previously queried vectors may be computed as the linear combination of previously returned keys. In more detail, say that $\mathsf{sk}_{\boldsymbol{y}} \in \mathbb{Z}^m$.[7] Say that the adversary queries vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k \in \mathbb{Z}_p^\ell$ and then submits a query $\boldsymbol{y}$ such that $\boldsymbol{y} = \sum_{j \in [k]} k_j \cdot \boldsymbol{y}_j \pmod p, \forall k_j \in \mathbb{Z}_p$. Then, the secret key $\mathsf{sk}_{\boldsymbol{y}}$ is computed as $\mathsf{sk}_{\boldsymbol{y}} = \sum_{j \in [k]} k_j \cdot \mathsf{sk}_{\boldsymbol{y}_j} \in \mathbb{Z}^m$. This property is satisfied by the LWE-based construction that evaluates inner products over $\mathbb{Z}_p$ in [8, Sec 4.2].

In the description hereunder, we assume that the modulus p is prime. However, the construction can also be applied to the Paillier-based construction of [8, Section 5.2], which evaluates inner products over $\mathbb{Z}_N$. As a result, it provides a simulation-secure IPFE with stateful key generation for inner products over $\mathbb{Z}_N$,

[7] The precise ring in which $\mathsf{sk}_{\boldsymbol{y}}$ lives is not important. We choose this to be $\mathbb{Z}$ for concreteness and compatibility with [8].

whereas our scheme in Section 4 is stateless but computes inner products over $\mathbb{Z}$. When we switch to composite moduli $N = pq$, we need to take into account that $\mathbb{Z}_N$ is not a field when the simulator has to solve a linear system over $\mathbb{Z}_N$ in order to compute a dummy message. Fortunately, inversion over $\mathbb{Z}_N$ is always possible with overwhelming probability when factoring N is hard.

5.1 Construction

Below, we provide our construction of $\mathsf{AdSimIPFE}$.

$\mathsf{Setup}(1^\lambda, 1^\ell, p)$: Given the security parameter λ, the supported message and key lengths ℓ and a prime integer p, do the following:

1. Set $L = 2\ell$ and obtain $(\mathsf{IPFE.mpk}, \mathsf{IPFE.msk}) \leftarrow \mathsf{IPFE.Setup}(1^\lambda, 1^L, p)$.
2. Output $(\mathsf{mpk}, \mathsf{msk}) := (\mathsf{IPFE.mpk}, \mathsf{IPFE.msk})$.

$\mathsf{Keygen}(\mathsf{msk}, \boldsymbol{y}, \mathsf{st})$: Given the msk, a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in \mathbb{Z}_p^\ell$ to obtain a key and an internal state st, do the following:

1. Parse the master secret key as $\mathsf{msk} = \mathsf{IPFE.msk}$.
2. The internal state st contains tuples $\left(\widehat{\boldsymbol{y}}_j, \boldsymbol{y}_j, \mathsf{sk}_{\boldsymbol{y}_j}, r_j\right)$ for some $j \in [\ell - 1]$ corresponding to (a subset of the) key queries made so far. If no queries have been made before $\boldsymbol{y}$, st is empty.
3. If $|\mathsf{st}| = i - 1$ for $i \in [\ell - 1]$ and $\boldsymbol{y} = \sum_{j=1}^{i-1} k_j \cdot \boldsymbol{y}_j \pmod p$ for some $k_j \in \mathbb{Z}_p$, $j \in [i-1]$, set $\widehat{\boldsymbol{y}} = \sum_{j=1}^{i-1} k_j \cdot \widehat{\boldsymbol{y}}_j \pmod p$ and compute the secret key as $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}})$. Set $\mathsf{sk}_{\boldsymbol{y}} := \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}}$.
4. Else, if $|\mathsf{st}| = i - 1$ for some $i \in [\ell - 1]$, set $\boldsymbol{y}_i = \boldsymbol{y}$. Then, construct the extended vector $\widehat{\boldsymbol{y}}_i = (\boldsymbol{y}_i, \mathbf{e}_i, r_i) \in \mathbb{Z}_p^L$, where $\mathbf{e}_i \in \mathbb{Z}_p^{\ell-1}$ is the i-th canonical vector and $r_i \leftarrow \mathbb{Z}_p$ is chosen uniformly at random. Next, compute a secret key $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}}_i)$ and set $\mathsf{sk}_{\boldsymbol{y}} := \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i}$. Update the internal state as $\mathsf{st} \leftarrow \mathsf{st} \cup \{(\widehat{\boldsymbol{y}}_i, \boldsymbol{y}_i, \mathsf{sk}_{\boldsymbol{y}_i}, r_i)\}$.
5. Output the secret key $\mathsf{sk}_{\boldsymbol{y}}$.

$\mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x})$: Given the mpk and a message $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_p^\ell$ to encrypt, do the following:

1. Parse the master public key as $\mathsf{mpk} = \mathsf{IPFE.mpk}$.
2. Construct the extended vector $\widehat{\boldsymbol{x}} = (\boldsymbol{x}, \mathbf{0}, 0) \in \mathbb{Z}_p^L$, where $\mathbf{0} \in \mathbb{Z}_p^{\ell-1}$ is the all-zeroes vector.
3. Compute a ciphertext $\mathsf{IPFE.ct} \leftarrow \mathsf{IPFE.Encrypt}(\mathsf{IPFE.mpk}, \widehat{\boldsymbol{x}})$.
4. Output the ciphertext $\boldsymbol{c} := \mathsf{IPFE.ct}$.

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, \boldsymbol{c})$: Given mpk, a secret key $\mathsf{sk}_{\boldsymbol{y}}$ and a ciphertext $\boldsymbol{c}$, do the following:

1. Parse the secret key as $\mathsf{sk}_{\boldsymbol{y}} = \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}}$ and the ciphertext as $\boldsymbol{c} = \mathsf{IPFE.ct}$.
2. Compute and output $z = \mathsf{IPFE.Decrypt}(\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}}, \mathsf{IPFE.ct})$.

Correctness. The correctness of AdSimIPFE is implied by the correctness of the underlying IPFE scheme as follows. For a message vector $\boldsymbol{x}$ and the i-th linearly independent vector $\boldsymbol{y} \in \mathbb{Z}_p^\ell$, let $\widehat{\boldsymbol{x}} = (\boldsymbol{x}, \mathbf{0}, 0), \widehat{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{e}_i, r_i) \in \mathbb{Z}_p^L$. When $\boldsymbol{y} \in \mathbb{Z}_p^\ell$ is linearly dependent on the previously queried vectors $\{\boldsymbol{y}_j \in \mathbb{Z}_p^\ell\}_{j\in[i-1]}$ for some $i \in [\ell - 1]$, we have $\widehat{\boldsymbol{y}} = \sum_{j=1}^{i-1} k_j \cdot \widehat{\boldsymbol{y}}_j \pmod p = \sum_{j=1}^{i-1} k_j \cdot \big(\boldsymbol{y}_j, \mathbf{e}_j, r_j\big) \pmod p$.

The Decrypt algorithm takes mpk, $\mathsf{sk}_{\boldsymbol{y}} = \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}}$ and $\boldsymbol{c} = \mathsf{IPFE.ct}$ as input, where we have the following.

$$\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}})$$
$$\mathsf{IPFE.ct} \leftarrow \mathsf{IPFE.Encrypt}(\mathsf{IPFE.mpk}, \widehat{\boldsymbol{x}})$$

Hence, the correctness of IPFE decryption algorithm forces the output to be $\mathsf{IPFE.Decrypt}(\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}}, \mathsf{IPFE.ct}) = \langle \widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}} \rangle = \langle (\boldsymbol{x}, \mathbf{0}, 0), (\boldsymbol{y}, \mathbf{e}_i, r_i) \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}_p$ as desired.

Efficiency. The efficiency of AdSimIPFE is inherited from the efficiency of the underlying IPFE scheme. The ciphertext and secret key sizes grow proportionally to $L = 2\ell = O(\ell)$.

5.2 Proof of Security for AdSimIPFE

Theorem 3. *The* AdSimIPFE *scheme achieves adaptive simulation based security, as long as the underlying* IPFE *scheme satisfies full adaptive indistinguishability based security.*

Proof. We assume w.l.o.g. that $\mathcal{A}$ makes secret key queries for linearly independent vectors only. In particular, we assume that $\mathcal{A}$ issues secret key queries for Q_{pre} independent vectors in the *pre-challenge* phase, which we denote by $\boldsymbol{y}_1^{\mathsf{pre}}, \ldots, \boldsymbol{y}_{Q_{\mathsf{pre}}}^{\mathsf{pre}} \in \mathbb{Z}_p^\ell$ while the i-th vector for the *post-challenge* independent secret key query is denoted as $\boldsymbol{y}_i^{\mathsf{post}} \in \mathbb{Z}_p^\ell$ such that $i \in [\ell - 1] \setminus [Q_{\mathsf{pre}}]$. Note that this simplification implies that there are no repetition in the key queries. We denote by $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_\ell^\star) \in \mathbb{Z}_p^\ell$ the message chosen by $\mathcal{A}$ in the challenge phase.

The Simulator: To simulate the real world scheme, the simulator uses the following tuple of PPT algorithms: $(\mathsf{Setup}^\star, \mathsf{Keygen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{Keygen}_1^\star)$. Note that $\mathsf{Keygen}_0^\star$ and $\mathsf{Keygen}_1^\star$ denote the simulated key generation algorithms to answer secret key queries in the pre-challenge and post-challenge phases respectively. The simulator then proceeds as follows.

$\mathsf{Setup}^\star(1^\lambda, 1^\ell, p)$: This algorithm is identical to **Setup** except that the simulator also samples $r_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell - 1]$ and maintains the internal state as the set of tuples $\mathsf{st}^\star = \{(\cdot, \cdot, \cdot, r_i)\}_{i\in[\ell-1]}$. In particular, it outputs the key pair as $(\mathsf{mpk}^\star, \mathsf{msk}^\star) := (\mathsf{IPFE.mpk}, \mathsf{IPFE.msk})$ while keeping $\mathsf{st}^\star$ to itself.

$\mathsf{Keygen}_0^\star(\mathsf{msk}^\star, \boldsymbol{y}^{\mathsf{pre}}, \mathsf{st}^\star)$: This algorithm runs almost identical to **Keygen**. In particular, on input a pre-challenge vector $\boldsymbol{y}^{\mathsf{pre}}$, it does the following:

1. The internal state $\mathsf{st}^\star$ contains tuples $\left(\widehat{\boldsymbol{y}}_j^{\mathsf{pre}}, \boldsymbol{y}_j^{\mathsf{pre}}, \mathsf{sk}_{\boldsymbol{y}_j^{\mathsf{pre}}}, r_j\right)$ for some $j \in [\ell-1]$ corresponding to (a subset of the) key queries made so far. If no query has been made before $\boldsymbol{y}^{\mathsf{pre}}$, $\mathsf{st}^\star$ is empty.
2. If $|\mathsf{st}| = i-1$ for some $i \in [\ell-1]$, set $\boldsymbol{y}_i^{\mathsf{pre}} = \boldsymbol{y}^{\mathsf{pre}}$. Then, construct the extended vector $\widehat{\boldsymbol{y}}_i^{\mathsf{pre}} = (\boldsymbol{y}_i^{\mathsf{pre}}, \mathbf{e}_i, r_i) \in \mathbb{Z}_p^L$. Next, compute a secret key $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}}_i^{\mathsf{pre}})$ and set $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}} := \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}}$. Update the internal state as $\mathsf{st}^\star \leftarrow \Big(\mathsf{st}^\star \cup \big\{(\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}, \boldsymbol{y}_i^{\mathsf{pre}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}, r_i)\big\}\Big) \backslash \big\{(\cdot,\cdot,\cdot,r_i)\big\}$.
3. Output the simulated secret key as $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}$.

$\mathsf{Encrypt}^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \mathcal{V}, \{1^{|x_i^\star|}\}_{i=1}^{\ell}, \mathsf{st}^\star)$: This algorithm takes $\mathsf{mpk}^\star = \mathsf{IPFE.mpk}$, $\mathsf{msk}^\star$, the lengths $\{1^{|x_i^\star|}\}_{i\in[\ell]}$ of all coordinates of the challenge message $\boldsymbol{x}^\star$ as input, the internal state $\mathsf{st}^\star$ and a set

$$\mathcal{V} = \left\{ \left(\boldsymbol{y}_j^{\mathsf{pre}}, z_j^{\mathsf{pre}} = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j^{\mathsf{pre}} \rangle, \mathsf{sk}_{\boldsymbol{y}_j^{\mathsf{pre}}} \right)_{j \in [Q_{\mathsf{pre}}]} \right\}$$

containing all pre-challenge independent queries $\{\boldsymbol{y}_j^{\mathsf{pre}}\}_{j\in[Q_{\mathsf{pre}}]}$, the returned keys and the corresponding linear function evaluations $\{z_j^{\mathsf{pre}} = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j^{\mathsf{pre}} \rangle\}_{j\in[Q_{\mathsf{pre}}]}$ for the challenge message $\boldsymbol{x}^\star$. The challenge ciphertext $\boldsymbol{c}^\star$ is simulated as follows.

1. Letting $\boldsymbol{z}_{\mathsf{pre}} = (z_1^{\mathsf{pre}}, \ldots, z_{Q_{\mathsf{pre}}}^{\mathsf{pre}})^\top \in \mathbb{Z}_p^{Q_{\mathsf{pre}}}$, it computes an arbitrary solution $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_\ell)^\top \in \mathbb{Z}_p^\ell$ of the system $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \pmod p$, where

$$\mathbf{Y}_{\mathsf{pre}} = \left[\boldsymbol{y}_1^{\mathsf{pre}} || \ldots || \boldsymbol{y}_{Q_{\mathsf{pre}}}^{\mathsf{pre}}\right]^\top \in \mathbb{Z}_p^{Q_{\mathsf{pre}} \times \ell}.$$

   Note that $\bar{\boldsymbol{x}}$ can be obtained via Gaussian elimination over $\mathbb{Z}_p$.
2. Construct the extended message vector $\widehat{\bar{\boldsymbol{x}}} = (\bar{\boldsymbol{x}}, -\mathbf{r}, 1) \in \mathbb{Z}_p^L$, where $\mathbf{r} = (r_1, \ldots, r_{\ell-1}) \in \mathbb{Z}_p^{\ell-1}$.[8]
3. Compute a ciphertext $\mathsf{IPFE.ct} \leftarrow \mathsf{IPFE.Encrypt}(\mathsf{IPFE.mpk}, \widehat{\bar{\boldsymbol{x}}})$.
4. Output the simulated ciphertext $\boldsymbol{c}^* := \mathsf{IPFE.ct}$.

$\mathsf{Keygen}_1^\star(\mathsf{msk}^\star, \boldsymbol{y}_i^{\mathsf{post}}, z_i^{\mathsf{post}}, \mathsf{st}^\star)$: On input a post-challenge vector $\boldsymbol{y}_i^{\mathsf{post}} \in \mathbb{Z}_p^\ell$, where $i \in \{Q_{\mathsf{pre}}+1, \ldots, \ell-1\}$, the linear function evaluation $z_i^{\mathsf{post}} = \langle \boldsymbol{x}^\star, \boldsymbol{y}_i^{\mathsf{post}} \rangle$ for the challenge message $\boldsymbol{x}^\star \in \mathbb{Z}_p^\ell$ and internal state $\mathsf{st}^\star$, it does the following:

1. The internal state $\mathsf{st}^\star$ now contains Q_{pre} tuples of the form $\left(\widehat{\boldsymbol{y}}_j^{\mathsf{pre}}, \boldsymbol{y}_j^{\mathsf{pre}}, \mathsf{sk}_{\boldsymbol{y}_j^{\mathsf{pre}}}, r_j\right)$ and tuples of the form $\left(\widehat{\boldsymbol{y}}_k^{\mathsf{post}}, \boldsymbol{y}_k^{\mathsf{post}}, \mathsf{sk}_{\boldsymbol{y}_k^{\mathsf{post}}}, r_k\right)$ for some $k \in [\ell-1] \setminus [Q_{\mathsf{pre}}]$ corresponding to (a subset of the) post-challenge key queries made so far. If $i = Q_{\mathsf{pre}}+1$, then $\mathsf{st}^\star = \left\{ \left(\widehat{\boldsymbol{y}}_j^{\mathsf{pre}}, \boldsymbol{y}_j^{\mathsf{pre}}, \mathsf{sk}_{\boldsymbol{y}_j^{\mathsf{pre}}}, r_j \right)_{j \in [Q_{\mathsf{pre}}]}, (\cdot,\cdot,\cdot,r_k)_{k \in [\ell-1]\setminus[Q_{\mathsf{pre}}]} \right\}$.

[8] For readability, we denote $-\mathbf{r} = (-r_1, \ldots, -r_{\ell-1}) = \mathbf{r}' \in \mathbb{Z}_p^{\ell-1}$ such that $\mathbf{r} + \mathbf{r}' = \mathbf{0} \bmod p$.

2. Construct the extended vector $\widehat{\boldsymbol{y}}_i^{\mathsf{post}} = (\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, \Delta_i + r_i)$, where $\mathbf{e}_i \in \mathbb{Z}_p^{\ell-1}$ is the i-th canonical vector, and $\Delta_i = z_i^{\mathsf{post}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle$. Next, compute a secret key $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}}_i^{\mathsf{post}})$ and set $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}} := \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}}$. Update the internal state as $\mathsf{st}^\star \leftarrow \left(\mathsf{st}^\star \cup \left\{ \left(\widehat{\boldsymbol{y}}_i^{\mathsf{post}}, \boldsymbol{y}_i^{\mathsf{post}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}, r_i\right) \right\}\right) \setminus \{(\cdot,\cdot,\cdot,r_i)\}$.
3. Output the simulated secret key as $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}$.

The Hybrids. We now prove that the simulation is computationally indistinguishable from the real experiment assuming full indistinguishability of IPFE. The proof proceeds via a sequence of games $(\mathbf{Game}_0, \mathbf{Game}_1, \mathbf{Game}_2, \mathbf{Game}_3)$. $\mathbf{Game}_0$ describes the interaction between the challenger and the adversary as in real experiment $\mathbf{Exp}^{\mathsf{Real}}_{\mathsf{AdSimIPFE},\mathcal{A}}(1^\lambda)$ while $\mathbf{Game}_3$ describes the same as in the ideal experiment $\mathbf{Exp}^{\mathsf{Ideal}}_{\mathsf{AdSimIPFE},\mathcal{A}}(1^\lambda)$.

In the following, let E_i denote the event that $\mathcal{A}$ wins in $\mathbf{Game}_i$. To prove the result, we will show that $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$ and $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda)$, which implies $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda)$.

$\mathbf{Game}_0$: In this game the challenger interacts with the adversary as in the real experiment.

$\mathbf{Game}_1$: In this game the setup phase is modified as follows. Beside computing $(\mathsf{mpk}, \mathsf{msk})$ as in the real experiment, the challenger now also precomputes $r_i \leftarrow \mathbb{Z}_p, \forall i \in [\ell - 1]$ for answering at most $\ell - 1$ linearly independent key queries as well as the challenge ciphertext query. It maintains an internal state $\mathsf{st}^\star = \{(\cdot,\cdot,\cdot,r_i)\}_{i\in[\ell-1]}$.

$\mathbf{Game}_2$: In this game the challenger changes the way the post-challenge keys are generated. It generates the pre-challenge keys as in $\mathbf{Game}_1$ with the precomputed randomness in $\mathsf{st}^\star$. It also generates the challenge ciphertext as before. As for post-challenge queries, they are answered as follows.

1. The challenger first computes $\bar{\boldsymbol{x}} \in \mathbb{Z}_p^\ell$ that is consistent with the Q_{pre} key vectors it encountered in the pre-challenge phase. In particular, letting $\boldsymbol{z}_{\mathsf{pre}} = (z_1^{\mathsf{pre}}, \ldots, z_{Q_{\mathsf{pre}}}^{\mathsf{pre}})^\top \in \mathbb{Z}_p^{Q_{\mathsf{pre}}}$ corresponding to the function evaluations $\{z_j^{\mathsf{pre}}\}_{j\in[Q_{\mathsf{pre}}]}$ on pre-challenge keys, it computes $\bar{\boldsymbol{x}} \in \mathbb{Z}_p^\ell$ such that

$$\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} (\mathrm{mod}\ p), \text{ where } \mathbf{Y}_{\mathsf{pre}} = \left[\boldsymbol{y}_1^{\mathsf{pre}} || \ldots || \boldsymbol{y}_{Q_{\mathsf{pre}}}^{\mathsf{pre}}\right]^\top \in \mathbb{Z}_p^{Q_{\mathsf{pre}} \times \ell}.$$

2. For all $i \in [\ell - 1] \setminus [Q_{\mathsf{pre}}]$, the i-th post-challenge vector $\boldsymbol{y}_i^{\mathsf{post}}$ is now extended as $\widehat{\boldsymbol{y}}_i^{\mathsf{post}} = (\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, \Delta_i + r_i)$, where $\Delta_i = z_i^{\mathsf{post}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle$.
3. The secret key is computed as $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}}_i^{\mathsf{post}})$.
4. The internal state is updated as

$$\mathsf{st}^\star \leftarrow \left(\mathsf{st}^\star \cup \left\{ \left(\widehat{\boldsymbol{y}}_i^{\mathsf{post}}, \boldsymbol{y}_i^{\mathsf{post}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}, r_i\right) \right\}\right) \setminus \{(\cdot,\cdot,\cdot,r_i)\}.$$

Game_3: In this game the challenger computes everything as before in Game_2 except that the challenge ciphertext is modified as follows. Instead of encrypting the extended message $\widehat{\boldsymbol{x}}^\star = (\boldsymbol{x}^\star, \mathbf{0}, 0)$, the challenger now encrypts $\widehat{\bar{\boldsymbol{x}}} = (\bar{\boldsymbol{x}}, -\mathbf{r}, 1)$ to compute $\boldsymbol{c}^\star := \mathsf{IPFE.ct} \leftarrow \mathsf{IPFE.Encrypt}(\mathsf{IPFE.mpk}, \widehat{\bar{\boldsymbol{x}}})$.

We now prove the following lemmas in order to complete the proof.

Lemma 7. *We have* $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

Proof. The change introduced here is only conceptual, where for all $i \in [\ell - 1]$, the randomness $r_i \in \mathbb{Z}_p$ are precomputed in the setup phase. Thus, the lemma follows trivially. □

Lemma 8. *We have* $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$.

Proof. We note that Game_2 only differs from Game_1 in the treatment of post-challenge queries. Specifically, the simulator $\mathcal{B}$ simulates $\mathcal{A}$'s view in the two games as follows.

1. On input $(1^\lambda, 1^\ell, p)$ from $\mathcal{A}$, $\mathcal{B}$ sets $L = 2\ell$ and computes $\mathsf{IPFE.Setup}(1^\lambda, 1^L, p)$ to obtain $(\mathsf{IPFE.mpk}, \mathsf{IPFE.msk})$. It sets $(\mathsf{mpk}, \mathsf{msk}) = (\mathsf{IPFE.mpk}, \mathsf{IPFE.msk})$, computes $r_i \leftarrow \mathbb{Z}_p$, for all $i \in [\ell - 1]$ to maintain its internal state as $\mathsf{st}^\star = \{(\cdot, \cdot, \cdot, r_i)\}_{i \in [\ell-1]}$. Finally, it sends mpk to $\mathcal{A}$.
2. When $\mathcal{A}$ requests a pre-challenge key for $\boldsymbol{y}_i^{\mathsf{pre}}$, $\mathcal{B}$ first computes the extended vector $\widehat{\boldsymbol{y}}_i^{\mathsf{pre}} = (\boldsymbol{y}_i^{\mathsf{pre}}, \mathbf{e}_i, r_i) \in \mathbb{Z}_p^L$, where $\mathbf{e}_i \in \mathbb{Z}_p^{\ell-1}$ is the i-th canonical vector. Using $\mathsf{msk} = \mathsf{IPFE.msk}$, it then obtains a secret key for $\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}$ as $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}}_i^{\mathsf{pre}})$. $\mathcal{B}$ then updates $\mathsf{st}^\star \leftarrow \left(\mathsf{st}^\star \cup \left\{(\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}, \boldsymbol{y}_i^{\mathsf{pre}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}, r_i)\right\}\right) \setminus \left\{(\cdot, \cdot, \cdot, r_i)\right\}$, sets $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}} = \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}}$ and sends $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ requests a challenge ciphertext for a message $\boldsymbol{x}^\star \in \mathbb{Z}_p^\ell$, $\mathcal{B}$ first computes an extended message $\widehat{\boldsymbol{x}}^\star = (\boldsymbol{x}^\star, \mathbf{0}, 0) \in \mathbb{Z}_p^L$, where $\mathbf{0} \in \mathbb{Z}_p^{\ell-1}$ is the all-zero vector. Using $\mathsf{mpk} = \mathsf{IPFE.mpk}$, it then obtains a ciphertext as $\mathsf{IPFE.ct} \leftarrow \mathsf{IPFE.Encrypt}(\mathsf{IPFE.mpk}, \widehat{\boldsymbol{x}}^\star)$, sets $\boldsymbol{c}^\star = \mathsf{IPFE.ct}$ and sends $\boldsymbol{c}^\star$ to $\mathcal{A}$.
4. In the post-challenge phase, when $\mathcal{A}$ queries a key for a vector $\boldsymbol{y}_i^{\mathsf{post}} \in \mathbb{Z}_p^\ell$, with $i \in [\ell - 1] \setminus [Q_{\mathsf{pre}}]$, for which the corresponding function evaluation is $z_i^{\mathsf{post}} = \langle \boldsymbol{x}^\star, \boldsymbol{y}_i^{\mathsf{post}} \rangle$, the challenger $\mathcal{B}$ responds as follows:
   - To simulate $\mathcal{A}$'s view in Game_1, $\mathcal{B}$ computes the extended vector $\widehat{\boldsymbol{y}}_i^{\mathsf{post}} = (\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, r_i)$.
   - To simulate $\mathcal{A}$'s view in Game_2, $\mathcal{B}$ first computes $\bar{\boldsymbol{x}} \in \mathbb{Z}_p^\ell$ as described in Game_2 such that $\bar{\boldsymbol{x}}$ is consistent the Q_{pre} pre-challenge key vectors. It then extends the vector $\boldsymbol{y}_i^{\mathsf{post}}$ as $\widehat{\boldsymbol{y}}_i^{\mathsf{post}} = (\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, \Delta_i + r_i)$, where $\Delta_i = \langle \boldsymbol{x}^\star - \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle$.

   Using $\mathsf{msk} = \mathsf{IPFE.msk}$, $\mathcal{B}$ obtains a secret key for $\widehat{\boldsymbol{y}}_i^{\mathsf{post}}$ as $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}} \leftarrow \mathsf{IPFE.Keygen}(\mathsf{IPFE.msk}, \widehat{\boldsymbol{y}}_i^{\mathsf{post}})$ and sets $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}} = \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}}$. It then updates

its internal state as $\mathsf{st}^\star \leftarrow \Big(\mathsf{st}^\star \cup \big\{(\widehat{\boldsymbol{y}}_i^{\mathsf{post}}, \boldsymbol{y}_i^{\mathsf{post}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}, r_i)\big\}\Big) \setminus \big\{(\cdot,\cdot,\cdot,r_i)\big\}$ and sends $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}$ to $\mathcal{A}$.

Recall that the only change between the two games is the way post-challenge keys are generated. In particular, the last co-ordinate of the i-th post-challenge key vector is set to r_i in $\mathbf{Game}_1$ while it is set to $(r_i + \Delta_i)$ in $\mathbf{Game}_2$. Note that each r_i is chosen uniformly in $\mathbb{Z}_p$ in the setup phase and is unique for each post challenge key query $i \in [\ell-1] \setminus [Q_{\mathsf{pre}}]$. Hence, the computation $r_i + \Delta_i$ being done modulo p, it follows that the two distributions $\{r_i \mid r_i \leftarrow \mathbb{Z}_p\}_{i\in[\ell-1]\setminus[Q_{\mathsf{pre}}]}$ and $\{r_i + \Delta_i \mid r_i \leftarrow \mathbb{Z}_p, \Delta_i \in \mathbb{Z}_p\}_{i\in[\ell-1]\setminus[Q_{\mathsf{pre}}]}$ are perfectly indistinguishable.

Further, any post-challenge key $\mathsf{sk}_{\boldsymbol{y}^{\mathsf{post}}}$ in $\mathbf{Game}_2$ always correctly decrypts any honestly generated ciphertext because such a ciphertext contains $\mathbf{0} \in \mathbb{Z}_p^\ell$ in its extended slots, which nullifies the extended slots of the keys. The two games are thus perfectly indistinguishable, which implies $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$. □

Lemma 9. *We have* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda)$.

Proof. Let us assume that $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]|$ is non-negligible. We then construct an adversary $\mathcal{B}$ that breaks the indistinguishability-based security of the underlying IPFE scheme as follows:

1. On input $(1^\lambda, 1^\ell, p)$ from $\mathcal{A}$, $\mathcal{B}$ sets $L = 2\ell$ and relays $(1^\lambda, 1^L, p)$ to the IPFE challenger. Upon receiving $\mathsf{IPFE.mpk}$, it sets $\mathsf{mpk}^\star = \mathsf{IPFE.mpk}$ and randomly chooses $r_i \leftarrow \mathbb{Z}_p$ for all $i \in [\ell-1]$ to maintain the internal state as $\mathsf{st}^\star = \{(\cdot,\cdot,\cdot,r_i)\}_{i\in[\ell-1]}$. It sends $\mathsf{mpk}^\star$ to $\mathcal{A}$.
2. When $\mathcal{A}$ requests a pre-challenge key for $\boldsymbol{y}_i^{\mathsf{pre}}$, $\mathcal{B}$ computes the extended vector $\widehat{\boldsymbol{y}}_i^{\mathsf{pre}} = (\boldsymbol{y}_i^{\mathsf{pre}}, \mathbf{e}_i, r_i) \in \mathbb{Z}_p^L$, where $\mathbf{e}_i \in \mathbb{Z}_p^{\ell-1}$ is the i-th canonical vector. It then queries the IPFE challenger with $\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}$ for a secret key and receives $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}}$. Then, $\mathcal{B}$ updates $\mathsf{st}^\star \leftarrow \Big(\mathsf{st}^\star \cup \big\{(\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}, \boldsymbol{y}_i^{\mathsf{pre}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}, r_i)\big\}\Big) \setminus \big\{(\cdot,\cdot,\cdot,r_i)\big\}$, sets $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}} = \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{pre}}}$ and sends $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ requests a challenge ciphertext, $\mathcal{B}$ sets $\boldsymbol{z}_{\mathsf{pre}} = (z_1^{\mathsf{pre}}, \ldots, z_{Q_{\mathsf{pre}}}^{\mathsf{pre}})^\top \in \mathbb{Z}_p^{Q_{\mathsf{pre}}}$ and then computes an arbitrary solution $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_\ell)^\top \in \mathbb{Z}_p^\ell$ of the linear system $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} (\mathrm{mod}\ p)$, where
$$\mathbf{Y}_{\mathsf{pre}} = \big[\boldsymbol{y}_1^{\mathsf{pre}} || \ldots || \boldsymbol{y}_{Q_{\mathsf{pre}}}^{\mathsf{pre}}\big]^\top \in \mathbb{Z}_p^{Q_{\mathsf{pre}} \times \ell}.$$
Next, it constructs the extended message vector $\widehat{\bar{\boldsymbol{x}}} = (\bar{\boldsymbol{x}}, -\mathbf{r}, 1) \in \mathbb{Z}_p^L$, where $\mathbf{r} = (r_1, \ldots, r_{\ell-1}) \in \mathbb{Z}_p^{\ell-1}$ to output $\boldsymbol{x}_0 = (\boldsymbol{x}^\star, \mathbf{0}, 0) \in \mathbb{Z}_p^L$ and $\boldsymbol{x}_1 = \widehat{\bar{\boldsymbol{x}}}$ as the pair of challenge messages to the IPFE challenger. The latter returns a challenge ciphertext $\mathsf{IPFE.ct}$ and $\mathcal{B}$ sets $\boldsymbol{c}^\star = \mathsf{IPFE.ct}$, which is returned to $\mathcal{A}$.
4. When $\mathcal{A}$ requests for a post-challenge key for $\boldsymbol{y}_i^{\mathsf{post}} \in \mathbb{Z}_p^\ell, i \in [\ell-1] \setminus [Q_{\mathsf{pre}}]$ with its corresponding function evaluation z_i^{post}, $\mathcal{B}$ computes $\Delta_i = z_i^{\mathsf{post}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}} \rangle$ and the extended vector $\widehat{\boldsymbol{y}}_i^{\mathsf{post}} = (\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, \Delta_i + r_i)$. It then queries the IPFE challenger with $\widehat{\boldsymbol{y}}_i^{\mathsf{post}}$ for a secret key and receives $\mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}}$. Then, $\mathcal{B}$ updates
$$\mathsf{st}^\star \leftarrow \Big(\mathsf{st}^\star \cup \Big\{ \Big(\widehat{\boldsymbol{y}}_i^{\mathsf{post}}, \boldsymbol{y}_i^{\mathsf{post}}, \mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}, r_i\Big) \Big\}\Big) \setminus \big\{(\cdot,\cdot,\cdot,r_i)\big\},$$

sets $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}} = \mathsf{IPFE.sk}_{\widehat{\boldsymbol{y}}_i^{\mathsf{post}}}$ and sends $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}$ to $\mathcal{A}$.

5. $\mathcal{B}$ outputs the same bit as $\mathcal{A}$.

Note that the ciphertext $\boldsymbol{c}^\star$ encodes the message $\boldsymbol{x}_0 = (\boldsymbol{x}^\star, \mathbf{0}, 0) \in \mathbb{Z}_p^L$ in $\mathbf{Game}_2$ and $\boldsymbol{x}_1 = \widehat{\bar{\boldsymbol{x}}} = (\bar{\boldsymbol{x}}, -\mathbf{r}, 1) \in \mathbb{Z}_p^L$ in $\mathbf{Game}_3$. The message $\bar{\boldsymbol{x}}$ in both games is computed maintaining the consistency with all the pre-challenge keys $\{\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{pre}}}\}_{i\in[Q_{\mathsf{pre}}]}$. Thus, upon decryption of $\boldsymbol{c}^\star$, it yields $\langle \boldsymbol{x}_0, \widehat{\boldsymbol{y}}_i^{\mathsf{pre}}\rangle = \langle \boldsymbol{x}^\star, \boldsymbol{y}_i^{\mathsf{pre}}\rangle = z^{\mathsf{pre}} \pmod p$ in $\mathbf{Game}_2$ as well as $\langle \boldsymbol{x}_1, \widehat{\boldsymbol{y}}_i^{\mathsf{pre}}\rangle = \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{pre}}\rangle + \langle -\mathbf{r}, \mathbf{e}_i\rangle + r_i = z^{\mathsf{pre}} \pmod p$ in $\mathbf{Game}_3$ as required. Further, note that in both games, for each $i \in [\ell - 1] \setminus [Q_{\mathsf{pre}}]$, the i-th post-challenge key $\mathsf{sk}_{\boldsymbol{y}_i^{\mathsf{post}}}$ is a secret key for the vector $\widehat{\boldsymbol{y}}_i^{\mathsf{post}} = (\boldsymbol{y}_i^{\mathsf{post}}, \mathbf{e}_i, \Delta_i + r_i)$, where

$$\Delta_i = z_i^{\mathsf{post}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}}\rangle,$$

which implies $\Delta_i + \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}}\rangle = z_i^{\mathsf{post}}$. Hence, upon decrypting $\boldsymbol{c}^\star$, we have

$$\langle \boldsymbol{x}_0, \widehat{\boldsymbol{y}}_i^{\mathsf{post}}\rangle = \langle \boldsymbol{x}^\star, \boldsymbol{y}_i^{\mathsf{post}}\rangle + \langle \mathbf{0}, \mathbf{e}_i\rangle + 0 \cdot (\Delta_i + r_i) = z_i^{\mathsf{post}} (\mathrm{mod}\ p),$$

in $\mathbf{Game}_2$, and

$$\langle \boldsymbol{x}_1, \widehat{\boldsymbol{y}}_i^{\mathsf{post}}\rangle = \langle \bar{\boldsymbol{x}}, \boldsymbol{y}_i^{\mathsf{post}}\rangle + \langle -\mathbf{r}, \mathbf{e}_i\rangle + 1 \cdot (\Delta_i + r_i) = z_i^{\mathsf{post}} (\mathrm{mod}\ p)$$

in $\mathbf{Game}_3$, as required. This proves that $\mathcal{B}$ is an admissible IPFE adversary in the indistinguishability-based security game. If the IPFE challenger returned a challenge ciphertext for the vector $\boldsymbol{x}_0$, $\mathcal{A}$'s view is as in $\mathbf{Game}_2$. Otherwise, $\mathcal{A}$'s view is the same as in $\mathbf{Game}_3$. Consequently, $\mathcal{B}$ breaks the adaptive indistinguishability-based security of the scheme if $\mathcal{A}$ can distinguish between the two games with noticeable advantage. □

Acknowledgements

Part of this work was funded by the French ANR ALAMBIC project (ANR-16-CE39-0006) and by BPI-France in the context of the national project RISQ (P141580). This work was also supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701).

References

1. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *Proceedings of PKC*, volume 9020 of *LNCS*, pages 733–751. Springer, 2015.
2. M Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Crypto*, 2018.
3. M Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *Eurocrypt*, 2017.

4. S. Agrawal, S. Bhattacherjee, D.-H. Phan, D. Stehlé, and S. Yamada. Efficient public trace and revoke from standard assumptions. In *ACM-CCS*, 2017.
5. S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *Proc. of PKC*, volume 7293 of *LNCS*, pages 280–297. Springer, 2012.
6. S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Proc. of Asiacrypt*, volume 7073 of *LNCS*, pages 21–40. Springer, 2011.
7. S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013.
8. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products from standard assumptions. In *Crypto*, 2016.
9. S. Agrawal and A. Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, 2017. `http://eprint.iacr.org/`.
10. Shweta Agrawal. Stronger security for reusable garbled circuits, new definitions and attacks. In *Crypto*, 2017.
11. F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC*, 2017.
12. A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *Asiacrypt 2015*, volume 9452 of *LNCS*, pages 470–491. Springer, 2015.
13. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput*, 32(3):586–615 (electronic), 2003.
14. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Eurocrypt*, 2014.
15. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
16. Z. Brakerski, A. Langlois, C. Peikert, Regev. O., and D. Stehlé. On the classical hardness of Learning With Errors. In *STOC*, 2013.
17. G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In *Asiacrypt*, 2018.
18. J. Chotard, E. Dufour Sans, R. Gay, D.-H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *Asiacrypt*, 2018.
19. P. Datta, R. Dutta, and S. Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC*, 2016.
20. P. Datta, T. Okamoto, and K. Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. In *Asiacrypt*, 2018.
21. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *PKC*, 2018.
22. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices and applications. In *Proc. of Eurocrypt*, LNCS. Springer, 2013.
23. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
24. S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Functional encryption without obfuscation. In *TCC (A2)*, pages 480–511. Springer, 2016.
25. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.
26. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Crypto*, 2012.

27. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC*, 2013.
28. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *Crypto*, 2015.
29. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM-CCS*, 2006.
30. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Eurocrypt*, 2008.
31. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Eurocrypt*, 2010.
32. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *Crypto*, 2017.
33. H. Lin and V. Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 2016.
34. T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *Asiacrypt*, 2009.
35. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Crypto*, 2010.
36. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *Eurocrypt*, 2012.
37. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`.
38. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of EUROCRYPT*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.
39. J. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13:433–447, 2000.
40. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
41. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Eurocrypt*, 2005.
42. A. Shamir. Identity-based cryptosystems and signature schemes. In *Crypto*, 1984.
43. J. Tomida, M. Abe, and T. Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In *ISC*, 2016.
44. J. Tomida and K. Takashima. Unbounded inner product functional encryption from bilinear maps. In *Asiacrypt*, 2018.
45. H. Wee. Dual system encryption via predicate encodings. In *TCC*, 2014.
46. H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *TCC*, 2017.