Adaptively Secure Permissive Unbounded Inner Product Functional Encryption from Lattices

Suprava Roy[1][0009−0006−0420−3632] and Ratna Dutta[1]

Department of Mathematics, Indian Institute of Technology Kharagpur, India
supravaroy9@kgpian.iitkgp.ac.in, ratna@maths.iitkgp.ac.in

Abstract. *Inner product functional encryption* (IPFE) in unbounded setting for permissive relation offers a powerful cryptographic approach for enforcing fine-grained access control over sensitive information in untrusted cloud settings and has found growing applicability in areas such as cloud security, controlled access to electronic health records, network privacy, mobile data protection, the Internet of Things and many more. In spite of its importance, the profile of this primitive is not extensive. In this work, we propose the *first permissive* and *unbounded* public key IPFE scheme, providing post-quantum security guarantees in privacy-preserving environments. Our construction is lattice-based, achieves adaptive security and builds on modified ALS-IPFE scheme of Abdalla et al. (ASIACRYPT 2020). The unbounded setting and permissive relation between message vector and secret key vector supported by our scheme make it more flexible and practical for adapting to real-world application. We establish the adaptive indistinguishability of our construction within the random oracle model, by providing formal proof assuming the hardness of the learning with errors problem. Our scheme is the first of its kind in the post-quantum setting as existing permissive unbounded IPFE are pairing-based and derive their security from classical number-theoretic assumptions. Furthermore, as exhibited by our experimental results our proposed scheme is computationally lightweight.

Keywords: Inner Product Functional Encryption · Unbounded Vectors · Permissive Relation · Adaptive Security · Learning with Errors

1 Introduction

Cloud Computing. The rapid advancement of digital technologies, accelerated by the global pandemic, has increased reliance on cloud computing and led to a substantial rise in data generation and network traffic. Organizations across various sectors, including education, healthcare and industry, utilize cloud-based platforms for storing, managing and transmitting large data volumes. This digital transformation enhances accessibility, scalability and efficiency, but poses significant challenges in security and privacy of sensitive data. The transmission and processing of highly confidential information, such as medical records and financial transactions, presents critical concerns in outsourced cloud environments with shared infrastructure accessed by multiple users and devices. Traditional

cryptographic systems, employing an all-or-nothing approach to data access, prove inadequate for the modern requirements of data sharing.

FE. Introduced by Boneh et al.[4], *functional encryption* (FE) provides a robust framework for fine-grained access control that surpasses the limitations of traditional public-key encryption. In an FE scheme for a function class $\mathfrak{F}$, a trusted authority distributes function-specific decryption keys dk_f for any function $f \in \mathfrak{F}$ derived from a master secret key. Decryption using dk_f reveals only the functional value $f(m)$ of the message m, ensuring that no information about the underlying message m is exposed beyond the functional output. Since its introduction, FE has emerged as a versatile cryptographic primitive, offering solutions to longstanding challenges in secure data processing and enabling privacy-preserving applications such as secure cloud computing, decentralized systems and advanced data analytics.

Sel-IND vs Adp-IND. The fundamental security mandate for FE is collusion resistance which ensures that an adversary possessing multiple functional decryption keys cannot deduce any information about an encrypted message beyond what is explicitly revealed by those keys. This is formally characterized by an indistinguishability-based security experiment in which a polynomial-time adversary $\mathcal{A}$ attempts to differentiate between two ciphertext instances, ct_{m_0} and ct_{m_1}, by querying a key extraction oracle. To avoid trivial success, the adversary is restricted to requesting decryption keys for functions f such that $f(m_0) = f(m_1)$. The distinction between security models is primarily temporal: in the *selective indistinguishability* (Sel-IND) model, the adversary is obligated to commit to its challenge messages m_0 and m_1 prior to the system's setup phase, whereas the *adaptive indistinguishability* (Adp-IND) model provides a significantly stronger security guarantee by permitting the adversary to select these messages dynamically at any point throughout the duration of the security game.

IPFE. *Inner product functional encryption* (IPFE) is a specialized variant of FE designed to support access control through inner product evaluations, making it ideal for applications involving vector-encoded data. In this setting, encryption associates a ciphertext with a data vector ($\boldsymbol{x}$) while decryption keys are linked to function vectors ($\boldsymbol{y}$) that determine the permitted computation. The concept of IPFE was first formalized by Abdalla et al.[1] who showed how a functional decryption key $\mathsf{dk}_{\boldsymbol{y}}$ corresponding to a vector $\boldsymbol{y}$ can be used to recover only the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ from a ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ encrypting a vector $\boldsymbol{x}$. Ciphertexts are generated using a master public key Mpk whereas functional decryption keys are derived from a master secret key Msk produced during an initial setup phase by a trusted authority. Their constructions achieve security in the Sel-IND model under standard cryptographic assumptions. Subsequent work by Agrawal et al.[3] strengthened these results by introducing adaptively secure IPFE schemes and constrained variants under similar assumptions.

Permissive uIPFE vs strict uIPFE. In bounded IPFE schemes, the size of the public parameters escalates in proportion to the magnitude of vectors, resulting in an unnecessary efficiency reduction. To address this limitation, the notion of *unbounded* IPFE (uIPFE) is garnering interest within the research community as

it allows the system parameters to be generated without prior knowledge of any vector length. Consequently, the sizes of the master public key and master secret key remain independent of the input vector dimensions. The unbounded variant of IPFE was independently introduced by Tomida et al.[12] and Dufour-Sans et al.[6] with constructions achieving IND security under pairing-based assumptions. There are two categories of uIPFE: *strict* and *permissive*. A strict uIPFE allows inner product evaluation when the message and secret key vectors share the same index set whereas the permissive uIPFE enables decryption as long as the index set of the secret key vector is contained within that of the message vector. Permissive uIPFE is well suited for a variety of privacy-critical applications including biometric matching based on Hamming distance, secure machine learning, medical data analytics and cloud-assisted computation. The following instance illustrates its practical relevance in controlled access of electronic medical records.

• Consider a cloud-enabled long-term healthcare monitoring scenario in which hospitals continuously collect patient data through connected medical devices. These devices generate sensitive measurements such as heart rate, blood pressure, oxygen saturation and glucose levels which are required for predictive analysis by cloud-based artificial intelligence (AI) services. Owing to strict privacy regulations, hospitals cannot disclose raw patient records and therefore encrypt the collected health data vectors ($\boldsymbol{x}$) using a uIPFE scheme before outsourcing the resulting ciphertexts ($\mathsf{ct}_{\boldsymbol{x}}$) to a public cloud. Authorized AI service providers request functional decryption keys ($\mathsf{dk}_{\boldsymbol{y}}$) from a trusted authority where the associated vector ($\boldsymbol{y}$) encodes the weights of an approved analytical or diagnostic model. Using these keys, the cloud can evaluate the corresponding inner product directly on encrypted data and obtain the required outputs ($\langle \boldsymbol{x}, \boldsymbol{y} \rangle$) without learning the underlying patient information ($\boldsymbol{x}$). The unbounded nature of uIPFE allows both data ($\boldsymbol{x}$) and model vectors ($\boldsymbol{y}$) to evolve over time which is essential in realistic medical settings where all relevant health attributes may not be known at the beginning of the treatment. Moreover, as new sensors are deployed and additional medical measurements are introduced, the permissive relation enables newly generated data components to be encrypted and appended to the existing encrypted data without re-encrypting existing records and without requiring prior knowledge of the final vector length. Since decryption is permitted whenever the index set of the function vector is contained within that of the data vector, analytical models can operate on selected subsets of features. In contrast, a strict uIPFE scheme would require the data and function vectors to share an identical index set, forcing re-encryption or key updates whenever new medical entities are added. As a result, permissive uIPFE provides the flexibility, scalability and efficiency required for applications involving continuous data generation and evolving analytical requirements.

Related Works. In 2015, Abdalla et al.[1] introduced the first IPFE scheme with Sel-IND security based on the Decisional Diffie-Hellman (DDH) assumption which was later strengthened by Agrawal et al.[3] in 2016 to achieve Adp-IND security under standard assumptions such as DDH, Decision Composite Residu-

osity (DCR) and Learning With Errors (LWE). The LWE-based IPFE from [3] is called ALS-IPFE. A modified ALS-IPFE was proposed by Abdalla et al.[2] in 2020 with Adp-IND security in *random oracle model* (ROM) under LWE assumption. Subsequent work has significantly expanded the IPFE framework, addressing features including stronger security notions, multi-user settings and unbounded vector scenarios. The concept of unbounded IPFE was independently proposed by Dufour-Sans et al.[6] and Tomida et al.[12] in 2018. Dufour-Sans et al.[6] presented two identity-based uIPFE schemes under strict and permissive relations with Sel-IND security while Tomida et al.[12] developed a private key IPFE and a public key IPFE under permissive relation, achieving Adp-IND security in the standard model without using random oracles. More recently, in 2025, Dutta et al.[7] introduced a post-quantum secure public-key uIPFE with Adp-IND security in the ROM under the LWE assumption along with a DDH-based Sel-IND secure private-key variant [1].

Our Contribution. Motivated by the practical importance of permissive uIPFE and emerging quantum threats, this work focuses on increasing the limited profile of permissive uIPFE. We construct an efficient post-quantum secure permissive uIPFE from lattices. To be more specific, we extend the modified ALS-IPFE [2] in permissive unbounded setting and present the *first* post-quantum secure permissive uIPFE construction. Our scheme achieves Adp-IND security under the LWE assumption in the ROM and its security is established through a rigorous formal proof. Transforming an IPFE to permissive uIPFE is not straightforward. We identify the challenges to make the modified ALS-IPFE [2] permissive and skillfully address these challenges to come up with our permissive uIPFE. We detail below a comparison evaluation of our proposed permissive uIPFE against existing permissive uIPFE constructions in terms of storage, computation complexity, communication overhead and security. As indicated in Table 1, the permissive public key uIPFE of Tomida et al. [12] achieves Adp-IND security based on the hardness of the SXDH problem. They also proposed a permissive uIPFE with Adp-IND security in private key setting that features function hiding in the sense that secret keys hide the information of the associated function. The permissive uIPFE scheme of Dufour-Sans et al.[6] exhibits Sel-IND security under the the Linearly Extended Decisional Bilinear Diffie-Hellman (ℓeDBDH) assumption. All these schemes derive their security from the classical number-theoretic assumptions and do not provide security against adversaries equipped with quantum computers. In contrast, our proposed scheme is the only lattice-based uIPFE construction that supports the permissive relation, offering post-quantum security. We emphasize that our scheme attains Adp-IND security and provides a stronger security guarantee than Sel-IND security of [6].

From the comparison Table 1, it is evident that our permissive uIPFE yields ciphertexts and functional decryption keys of linear size which grows according to

[1] We note that a lattice-based permissive uIPFE scheme under the LWE assumption in the standard model was recently proposed independently and concurrently by Cini and Tairi [5]. Their constructions further support identity-based and attribute-based access control mechanisms while achieving security in the standard model.

Table 1. An efficiency assessment of existing public key permissive uIPFE

uIPFE schemes	|Msk|	|ct|	|dk|	Assumtion	Security	ROM /SM	Post-quantum secure (PQS)
[6]	$\|\mathbb{G}_1\|+\|\mathbb{Z}_q\|$	$\|I\|\|\mathbb{G}\| + \|\mathbb{G}_1\|$	$\|I'\|\|\mathbb{G}\|+ \|\mathbb{G}_2\|$	ℓeDBDH	Sel-IND	ROM	No
[12]	$28 \cdot \|\mathbb{Z}_q\|$	$7\|I\|\|\mathbb{G}_1\|$	$7\|I'\|\|\mathbb{G}_2\| + \|I'\|$	SXDH	Adp-IND	SM	No
Our	$m^2 \cdot \|\mathbb{Z}\|$	$(\|I\|+m)\ \|\mathbb{Z}_q\|$	$(\|I'\|+m)\ \|\mathbb{Z}_q\|$	LWE	Adp-IND	ROM	Yes

ℓeDBDH : linearly extended decisional bilinear Diffie-Hellman
SXDH : symmetric external Diffie-Hellman
LWE : learning with errors
Sel-IND: selective indistinguishability security
Adp-IND: adaptive indistinguishability security

ROM : random oracle model
SM: standard model
$|\mathbb{Z}|$: size of an element in $\mathbb{Z}$
$|\mathbb{G}|, |\mathbb{G}_1|, |\mathbb{G}_2|$: size of an element of the groups $\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2$

|Msk| : size of master secret key
|ct| : size of ciphertext
|dk| : size of decryption key
$|I|$: dimension of message vectors
$|I'|$: dimension of secret key vectors
q : a large prime
m : a positive integer
$|\mathbb{Z}_q|$: size of an element in $\mathbb{Z}_q$

the size of message vectors and secret key vectors respectively while maintaining a constant-size master secret key. While the public key in [12] consists of a single group element and that of [6] comprises 28 group elements, our scheme achieves a constant-size master public key, represented by a matrix in $\mathbb{Z}_q$ of dimension $n \times m$.

Table 2. A comparative analysis of computation cost of existing permissive public key uIPFE

uIPFE schemes	# pair	# mult	# expo	# hash
[6]	$2\|I\| + 1$	$2\|I\| + \|I'\| + 1$	$(2\|I\| + \|I'\| + 2)$	$\|I\| + \|I'\|$
[12]	$\|I'\|$	$49\|I\| + 49\|I'\|$	$7\|I\| + 7\|I'\| + 28$	0
Our	0	$mn + m + (n + 1)(\|I\| + \|I'\|)$	0	$\|I\| + \|I'\|$

pair: number of pairings
mult: number of multiplications
$|I|$: dimension of the message vector
$|I'|$: dimension of the secret key vector

expo: number of exponentiations
hash: number of hash function calls
n, m: order of matrix

Table 2 presents a comparative analysis of computation cost of existing permissive uIPFE schemes for encryption and decryption. Unlike the existing schemes, our construction avoids the use of expensive bilinear pairings and exponentiation operations. Instead, it uses matrix multiplication and evaluation of hash function, making it efficient computationally. Pairing-based operations generally involve expensive computations over large extension fields including elliptic curve multiplications and final exponentiations, eliminating these operations results in significantly improved computational efficiency. In addition, the setup and key extraction phases of our scheme use two polynomial-time algorithms

TrapdoorGen and SamplePre, each of these algorithms takes $O(n^2 \log q)$ time. During decryption, the noise is minimized by performing an exhaustive search over the set $\{0, 1, \ldots, K-1\}$. While this exhaustive search carries a theoretical complexity of $O(K)$, our implementation confirms its practical efficiency. For instance, with K set to $65,536$ (2^{16}), decryption requires approximately 0.810 ms. This performance is well-suited for cloud-based, privacy-preserving tasks such as biometric matching or aggregate statistics. Furthermore, because linear searches are highly parallelizable, this computational cost remains negligible in high-throughput systems compared to network latency and standard lattice-based cryptographic operations.

In sum, our permissive uIPFE scheme is computationally lightweight and flexible as compared to existing similar schemes. Additionally, by supporting decryption under permissive relation between message and function vectors, it offers greater practicality and adaptability for real-world applications compared to the LWE-based adaptively secure strict uIPFE construction of Dutta et al.[7]. Additionally, we implement our proposed permissive uIPFE scheme in the C programming language and evaluate its performance on an IBM Power9 server. For a specific choice of parameters, we report representative performance results of the scheme. Table 3 summarizes the evaluation time of each algorithm for varying message vector lengths from which it is evident that our proposed scheme is computationally lightweight.

Table 3. Performance of our permissive uIPFE for different lengths (ℓ) of message vectors with $\ell = 128, 256, 512, 1024, 2048$ using parameters $n = 5$, $q \approx 2^{49}$, $m = 490$, $\ell_{\mathsf{max}} = 4096$, $\psi = 2.236$, $\alpha = 4.495 \times 10^{-14}$, $\sigma = 83.29$, $X = 4$ and $Y = 4$.

message vector lerngth (ℓ)	128	256	512	1024	2048
uIPFE.KeyExtract	1.993 sec	3.197 sec	5.701 sec	10.454 sec	20.554 sec
uIPFE.Encrypt	4.390 ms	4.928 ms	7.039 ms	11.061 ms	17.662 ms
uIPFE.Decrypt	0.058 ms	0.081 ms	0.119 ms	0.203 ms	0.499 ms

Technical Overview. A brief overview of our permissive uIPFE is presented below which is an enhancement of the modified ALS-IPFE [2] in the sense that $I_{\boldsymbol{y}} = I_{\boldsymbol{x}}$ in [2] whereas in our construction allows $I_{\boldsymbol{y}} \subseteq I_{\boldsymbol{x}}$.
In setup phase, a trusted authority sets the system parameter $n, m, q, \ell_{\mathsf{max}}, X, Y, K = XY\ell_{\mathsf{max}}, \sigma, \alpha, \psi$ and runs the polynomial time trapdoor generation algorithm $\mathsf{TrapdoorGen}(n,\ m, q) \rightarrow (\mathbf{A}, \mathbf{T})$ to produce a parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ that is statistically indistinguishable from uniform, paired with a trapdoor basis $\mathbf{T} \in \mathbb{Z}^{m \times m}$ satisfying $\mathbf{AT} = \mathbf{0}(\bmod q)$. A full domain hash function $\mathcal{H} : \mathbb{N} \rightarrow \mathbb{Z}_q^n$ is chosen which is modeled as a random oracle. The authority then sets the master public key $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$ and the master secret

key $\mathsf{Msk} = \mathbf{T}$.
In key extraction phase, the trusted authority generates a decryption key $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \boldsymbol{y}^\top \mathbf{Z}^\top)$ corresponding to a secret key vector $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}} \in \mathbb{Z}_q^{|I_{\boldsymbol{y}}|}$ where $\mathbf{Z} \in \mathfrak{D}_{\mathbb{Z}^{m \times |I_{\boldsymbol{y}}|}, \sigma}$ is generated by the polynomial time algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}') \to \mathbf{Z}$ with $\mathbf{U}' = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{|I_{\boldsymbol{y}}|}) \in \mathbb{Z}_q^{n \times |I_{\boldsymbol{y}}|}$, $\boldsymbol{u}'_k = \mathcal{H}(i'_k) \in \mathbb{Z}_q^n$, $i'_k \in I_{\boldsymbol{y}}$, master secret key $\mathbf{T}$, discrete Gaussian parameter σ satisfying $\mathbf{AZ} = \mathbf{U}'$ and sends $\mathsf{dk}_{\boldsymbol{y}}$ to a user.
In encryption phase, a user using the master public key Mpk encrypts a message vector $\boldsymbol{x} = (x_i)_{i \in I_{\boldsymbol{x}}} \in \mathbb{Z}_q^{|I_{\boldsymbol{x}}|}$ and generates the ciphertext $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ corresponding to the message vector $\boldsymbol{x}$ where $\boldsymbol{c}_1 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}_1 \in \mathbb{Z}_q^m$, $\boldsymbol{c}_2 = \mathbf{U}^\top \boldsymbol{s} + \boldsymbol{e}_2 + \lfloor \frac{q}{K} \rfloor \boldsymbol{x} \in \mathbb{Z}_q^{|I_{\boldsymbol{x}}|}$ with $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{|I_{\boldsymbol{x}}|}) \in \mathbb{Z}_q^{n \times |I_{\boldsymbol{x}}|}$, $\boldsymbol{u}_k = \mathcal{H}(i_k) \in \mathbb{Z}_q^n$, $i_k \in I_{\boldsymbol{x}}$ and $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_1 \in \mathfrak{D}_{\mathbb{Z}^m, \psi}$, $\boldsymbol{e}_2 \in \mathfrak{D}_{\mathbb{Z}^{|I_{\boldsymbol{x}}|}, \psi}$.
The decryptor decrypts the ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ using the decryption key $\mathsf{dk}_{\boldsymbol{y}}$ and recovers the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. The main challenge the decryptor faces is that the dimentions of $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}} \in \mathbb{Z}_q^{|I_{\boldsymbol{y}}|}$ and $\boldsymbol{c}_2 \in \mathbb{Z}_q^{|I_{\boldsymbol{x}}|}$ are different. To address this issue, we observe that the decryptor computes $\mu' = \sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} c_{2,k} - (\boldsymbol{y}^\top \mathbf{Z}^\top) \boldsymbol{c}_1$ where $c_{2,k}$ being the k-th coefficient of $\boldsymbol{c}_2$. Then correctness holds as

$$\begin{aligned}
\mu' &= \sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} c_{2,k} - (\boldsymbol{y}^\top \mathbf{Z}^\top) \boldsymbol{c}_1 \\
&= \boldsymbol{y}^\top \mathbf{U}'^\top \boldsymbol{s} + \boldsymbol{y}^\top \boldsymbol{e}_2 + \left\lfloor \frac{q}{K} \right\rfloor \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \boldsymbol{y}^\top \mathbf{U}'^\top \boldsymbol{s} - \boldsymbol{y}^\top \mathbf{Z}^\top \boldsymbol{e}_1 \\
&= \left\lfloor \frac{q}{K} \right\rfloor \langle \boldsymbol{x}, \boldsymbol{y} \rangle + \mathsf{noise}
\end{aligned}$$

This is possible as $\boldsymbol{u}_k = \boldsymbol{u}'_k$ for all $i_k \in I_{\boldsymbol{x}} \cap I_{\boldsymbol{y}}$ where $\boldsymbol{u}_k = \mathcal{H}(i_k)$ and $\boldsymbol{u}'_k = \mathcal{H}(i'_k)$ are the k-th column of $\mathbf{U}$ and $\mathbf{U}'$ respectively. Finally $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is recovered by eliminating the noise through exaustively searching values $\mu \in \{0, 1, \ldots, K-1\}$ until $|\lfloor \frac{q}{K} \rfloor \mu - \mu'| = \mathsf{noise} \leq \frac{q}{2K}$.
The decryption of our scheme yields accurate result provided the error quantity $\mathsf{noise} \leq Y\psi\ell_{\mathsf{max}} + \sigma\psi m\ell_{\mathsf{max}} Y \leq \frac{q}{2K}$ i.e. $q \geq 2K(Y\psi\ell_{\mathsf{max}} + \sigma\psi m\ell_{\mathsf{max}} Y)$.
We face another challenge in the security proof which is deduced from the $\mathsf{Adp\text{-}IND}$ security of modified $\mathsf{ALS\text{-}IPFE}$ [2]. In modified $\mathsf{ALS\text{-}IPFE}$, the queried secret key vectors $\boldsymbol{y}$ has the same length $|I_{\boldsymbol{y}}|$ as the length $|I_{\boldsymbol{x}}|$ of the message vectors $\boldsymbol{x}$. In contrast, the length of the queried secret key vectors $\boldsymbol{y}$ in our permissive uIPFE has length less than or equal to that of the message vectors $\boldsymbol{x}$. We overcome this issue by transforming $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}} \in \mathbb{Z}_q^{|I_{\boldsymbol{y}}|}$ into $\hat{\boldsymbol{y}} = (\hat{y}_i)_{i \in I_{\boldsymbol{x}}} \in \mathbb{Z}_q^{|I_{\boldsymbol{x}}|}$ by setting $\hat{y}_i = \begin{cases} y_i \text{ for all } i \in I_{\boldsymbol{y}} \\ 0 \text{ for all } i \in I_{\boldsymbol{x}} \setminus I_{\boldsymbol{y}} \end{cases}$ Besides, $\mathsf{Msk} = \mathbf{Z}_{\mathsf{ALS}} \in \mathfrak{D}_{\mathbb{Z}^{m \times |I_{\boldsymbol{x}}|}, \sigma}$ in modified $\mathsf{ALS\text{-}IPFE}$ while $\mathsf{Msk} = \mathbf{Z} \in \mathfrak{D}_{\mathbb{Z}^{m \times |I_{\boldsymbol{y}}|}, \sigma}$ in our permissive uIPFE. This forbids us to directly set $\mathbf{Z} = \mathbf{Z}_{\mathsf{ALS}}$ as $|I_{\boldsymbol{x}}| \neq |I_{\boldsymbol{y}}|$. In the security reduction, we observe that we can fix this problem by removing $(|I_{\boldsymbol{x}}| - |I_{\boldsymbol{y}}|)$ columns of $\mathbf{Z}_{\mathsf{ALS}}$ more precisely by deleting k-th column of $\mathbf{Z}_{\mathsf{ALS}}$ for all $i_k \in I_{\boldsymbol{x}} \setminus I_{\boldsymbol{y}}$.

In our work, we adopt the standard notion of unbounded IPFE as established by Tomida et al.[12] and Dufour-Sans et al.[6]. While the system parameters, specifically the modulus q and matrix dimension m are tied to a maximum capacity ℓ_{max} during the setup phase, this configuration remains static for all vectors with length ℓ where $\ell \leq \ell_{\mathsf{max}}$. Consequently, our scheme functions as an IPFE with a compact public key that supports flexible, variable-length inputs, eliminating the need to update parameters for different dimensions.

In practical scenarios, the length of an attribute vector, such as one representing a patient's record is often relatively small. One might assume that utilizing a standard bounded IPFE scheme, such as the original ALS-IPFE[3] and padding unused attributes with zeros up to a predefined maximum length would outperform our proposed unbounded construction. However, this approach introduces significant overhead. For instance, if a vector of length 100 is padded to a maximum length of $10,000$, the master public and secret keys in a bounded setting must scale proportionally with the padded dimension. In contrast, our scheme maintains constant-size master keys regardless of the vector length. Furthermore, zero-padding forces the generation of decryption keys and ciphertexts corresponding to the maximum length, leading to redundant computations and a substantial loss in efficiency for smaller inputs.

Organization. The remainder of this paper is structured as follows. Section 2 introduces the required notation, background concepts, underlying assumptions and relevant lemmas. The proposed post-quantum secure public-key permissive uIPFE construction, together with a comprehensive security analysis, is presented in Section 3. Lastly, Section 4 concludes the paper and outlines potential directions for future research.

2 Preliminaries

Notations We denote $\lambda \in \mathbb{N}$ as a security parameter and $\mathsf{poly}(\lambda)$ as a generic polynomial function in λ. For any prime q, we let $\mathbb{Z}_q$ represents the field $\mathbb{Z}/q\mathbb{Z}$. For a natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. Sampling an element s uniformly from a set or distribution S is represented by $s \xleftarrow{\$} S$. Vectors and matrices are denoted by bold lowercase and bold uppercase letters, respectively. For a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$, the ℓ_2-norm of $\boldsymbol{v}$ is defined as $\|\boldsymbol{v}\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$. Given a matrix $\mathbf{M} \in \mathbb{Z}_q^{m \times n}$ with columns $(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n)$, the max norm is defined as $\|\mathbf{M}\| = \mathsf{max}_{i \in [n]} \|\boldsymbol{m}_i\|$. The Gram-Schmidt orthogonalization of the columns of $\mathbf{M}$ is denoted by $\widetilde{\mathbf{M}}$. Finally, $\mathbf{M}^\top$ represents the transpose of $\mathbf{M}$, $\mathbf{I}_m$ denotes the m-dimensional identity matrix and $A \approx_s B$ indicates that two distributions A and B are statistically close.

Let Γ be an inner product space over a field F. We represent an unbounded vector $\boldsymbol{x} \in \Gamma$ as $\boldsymbol{x} = (x_i)_{i \in I_{\boldsymbol{x}}} \in \Gamma$ where $I_{\boldsymbol{x}}$ is a finite subset of $\mathbb{N}$ with $x_i = 0$ for $i \notin I_{\boldsymbol{x}}$. Given two such vectors, $\boldsymbol{x} = (x_i)_{i \in I_{\boldsymbol{x}}}$ and $\boldsymbol{y} = (y_j)_{j \in I_{\boldsymbol{y}}}$, the inner product is computed over the intersection of their active indices:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{k \in I_{\boldsymbol{x}} \cap I_{\boldsymbol{y}}} x_k y_k.$$

This comes from the fact that $x_k = 0$ implicitly for indices $k \notin I_{\boldsymbol{x}}$ and $y_k = 0$ implicitly for $k \notin I_{\boldsymbol{y}}$. A *probabilistic polynomial time* (PPT) algorithm is a probabilistic model run by a Turing machine in $\mathsf{poly}(\lambda)$ time. Furthermore, we define a negligible function $\mathsf{negl}(\cdot) : \mathbb{N} \to \mathbb{R}$ as a function that vanishes faster than the inverse of any polynomial. Formally, $\mathsf{negl}(n)$ is negligible if for every positive constant c, there exists an integer N such that for all $n \geq N$, the condition $\mathsf{negl}(n) \leq n^{-c}$ is satisfied.

Definition 21 (Permissive Relation) *[6] Let $\mathcal{X}$ and $\mathcal{Y}$ be two subsets of an inner product space Γ. A* permissive *relation $\mathcal{R}_p : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is defined by the set*

$$\mathcal{R}_p = \{(\boldsymbol{a} = (a_i)_{i \in I_{\boldsymbol{a}}}, \boldsymbol{b} = (b_i)_{i \in I_{\boldsymbol{b}}}) \in \mathcal{X} \times \mathcal{Y} \mid I_{\boldsymbol{b}} \subseteq I_{\boldsymbol{a}}\}.$$

For $\mathcal{R}_p(\boldsymbol{a}, \boldsymbol{b}) = 1$, the inner product $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i \in I} a_i b_i$ where $I = I_{\boldsymbol{a}} \cap I_{\boldsymbol{b}} = I_{\boldsymbol{b}}$.

Definition 22 (Strict Relation) *[6] Let $\mathcal{X} \subset \Gamma$ and $\mathcal{Y} \subset \Gamma$ where Γ is an inner product space . A* strict *relation $\mathcal{R}_s : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is the set*

$$\mathcal{R}_s = \{(\boldsymbol{a} = (a_i)_{i \in I_{\boldsymbol{a}}}, \boldsymbol{b} = (b_i)_{i \in I_{\boldsymbol{b}}}) \in \mathcal{X} \times \mathcal{Y} \mid I_{\boldsymbol{b}} = I_{\boldsymbol{a}}\}.$$

For $\mathcal{R}_s(\boldsymbol{a}, \boldsymbol{b}) = 1$, the inner product $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i \in I} a_i b_i$ where $I = I_{\boldsymbol{a}} = I_{\boldsymbol{b}}$.

Observe that $\mathcal{R}_s(\boldsymbol{a}, \boldsymbol{b}) = 1$ if $\mathcal{R}_p(\boldsymbol{a}, \boldsymbol{b}) = 1$ $\forall (\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{X} \times \mathcal{Y}$.

2.1 Permissive Unbounded Inner Product Functional Encryption [6]

Definition 23 *A* permissive unbounded inner product functional encryption *(uIPFE) over an inner product space Γ over a field F is a tuple of four PPT algorithms $\mathsf{uIPFE}_{permissive} = (\mathsf{Setup}, \mathsf{KeyExtract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with a space of message vectors $\mathcal{X}$, a space of secret key vectors $\mathcal{Y}$ and a permissive relation $\mathcal{R}_p : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ satisfying $\mathcal{R}_p = \{(\boldsymbol{a} = (a_i)_{i \in I_{\boldsymbol{a}}}, \boldsymbol{b} = (b_i)_{i \in I_{\boldsymbol{b}}}) \in \mathcal{X} \times \mathcal{Y} | I_{\boldsymbol{b}} \subseteq I_{\boldsymbol{a}}\}$ where $\mathcal{X}, \mathcal{Y} \subseteq \Gamma$ fulfilling the following requirements:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{Mpk}, \mathsf{Msk})$*: Given a security parameter λ, this algorithm, run by a trusted authority outputs a master public key Mpk and a master secret key Msk. The master public key Mpk is publicly available, while the master secret key Msk is kept private by the trusted authority itself.*
- $\mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y}) \to \mathsf{dk}_{\boldsymbol{y}}$*: The trusted authority generates a decryption key $\mathsf{dk}_{\boldsymbol{y}}$ containing $\boldsymbol{y} \in \mathcal{Y}$ for a decryptor using Mpk, Msk and a secret key vector $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}} \in \mathcal{Y}$ where $I_{\boldsymbol{y}} \subseteq \mathbb{N}$ is the finite index set of $\boldsymbol{y}$. The decryption key $\mathsf{dk}_{\boldsymbol{y}}$ is distributed to the intended decryptor through a secure channel.*
- $\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}) \to \mathsf{ct}_{\boldsymbol{x}}$*: The encryptor computes a ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ for a message vector $\boldsymbol{x} = (x_i)_{i \in I_{\boldsymbol{x}}} \in \mathcal{X}$ where $I_{\boldsymbol{x}} \subseteq \mathbb{N}$ represents the finite index set of $\boldsymbol{x}$. The resulting ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ is broadcast over an insecure public channel.*

– $\mathsf{Decrypt}(\mathsf{Mpk}, \mathsf{dk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) \rightarrow \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ *or* $\perp$*: The decryptor utilizes its decryption key* $\mathsf{dk}_{\boldsymbol{y}}$ *corresponds to the* $\boldsymbol{y} \in \mathcal{Y}$ *to decrypt a ciphertext* $\mathsf{ct}_{\boldsymbol{x}}$ *corresponding to the* $\boldsymbol{x} \in \mathcal{X}$ *using* Mpk*. If the permissive relation* $\mathcal{R}_p(\boldsymbol{x}, \boldsymbol{y}) = 1$ *holds, the algorithm recovers the inner product* $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$*; otherwise it outputs a failure symbol* $\perp$*.*

Correctness: A permissive unbounded inner product functional encryption scheme $\mathsf{uIPFE}_{\mathsf{permissive}} = (\mathsf{Setup}, \mathsf{KeyExtract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is considered *correct* if, for every security parameter $\lambda \in \mathbb{N}$, any message vector $\boldsymbol{x} = (x_i)_{i \in I_{\boldsymbol{x}}} \in \mathcal{X}$ and any secret key vector $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}} \in \mathcal{Y}$ that satisfies the permissive relation $\mathcal{R}_p(\boldsymbol{x}, \boldsymbol{y}) = 1$, the following holds:

$$\Pr\left[\mu = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i \in I_{\boldsymbol{x}} \cap I_{\boldsymbol{y}}} x_i y_i \;\middle|\; \begin{array}{l} (\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Setup}(1^\lambda); \\ \mathsf{dk}_{\boldsymbol{y}} \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y}); \\ \mathsf{ct}_{\boldsymbol{x}} \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}); \\ \mu \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Decrypt}(\mathsf{Mpk}, \mathsf{dk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) \end{array}\right] = 1$$

Definition 24 (Adaptive indistinguishability (Adp-IND)) *A permissive unbounded inner product functional encryption* $\mathit{uIPFE}_{\mathit{permissive}} = (\mathit{Setup}, \mathit{KeyExtract}, \mathit{Encrypt}, \mathit{Decrypt})$ *is defined as adaptive indistinguishable (Adp-IND) secure if no PPT adversary* $\mathcal{A}$ *can succeed in the security experiment with a non-negligible advantage over a random guess.*
Formally, for every PPT adversary $\mathcal{A}$ *and security parameter* $\lambda \in \mathbb{N}$*, the following advantage must be negligible:*

$$\mathsf{Adv}^{\mathit{uIPFE}_{\mathit{permissive}}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda) = |\Pr[\mathsf{Expt}^{\mathit{uIPFE}_{\mathit{permissive}}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

where the security experiment $\mathsf{Expt}^{\mathit{uIPFE}_{\mathit{permissive}}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda)$ *between a challenger* $\mathcal{C}$ *and the adversary* $\mathcal{A}$ *is described below in Figure 1 and* $\mathsf{negl}(\lambda)$ *is a negligible function in* λ*.*

2.2 Background Related to Lattices

Integer Lattices: [2] For a prime q and a parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define the *orthogonal q-ary lattice* as $\Lambda_q^{\perp}(\mathbf{A}) = \{\boldsymbol{r} \in \mathbb{Z}^m \mid \mathbf{A}\boldsymbol{r} \equiv \mathbf{0} \pmod q\}$. For a non-zero target vector $\boldsymbol{v} \in \mathbb{Z}_q^n$, we define the *shifted q-ary lattice* as: $\Lambda_q^{\boldsymbol{v}}(\mathbf{A}) = \{\boldsymbol{r} \in \mathbb{Z}^m \mid \mathbf{A}\boldsymbol{r} \equiv \boldsymbol{v} \pmod q\}$. For any target matrix $\mathbf{V} \in \mathbb{Z}_q^{n \times \ell}$, we define the generalized shifted lattice as the set of matrices $\mathbf{R} \in \mathbb{Z}_q^{m \times \ell}$ satisfying the following congruence: $\Lambda_q^{\mathbf{V}}(\mathbf{A}) = \{\mathbf{R} \in \mathbb{Z}_q^{m \times \ell} \mid \mathbf{A}\mathbf{R} \equiv \mathbf{V} \pmod q\}$.
Discrete Gaussian: [9] Let $\Lambda \subset \mathbb{Z}^m$ be an integer lattice. The *discrete Gaussian distribution* over Λ, parameterized by a width $\psi > 0$ and a center $\boldsymbol{c} \in \mathbb{R}^m$, is defined as: $\mathfrak{D}_{\Lambda,\psi,\boldsymbol{c}}(\boldsymbol{y}) = \frac{\exp\left(-\pi\|\boldsymbol{y}-\boldsymbol{c}\|^2/\psi^2\right)}{\sum_{\boldsymbol{x}\in\Lambda}\exp(-\pi\|\boldsymbol{x}-\boldsymbol{c}\|^2/\psi^2)}, \quad \forall \boldsymbol{y} \in \Lambda$. For convenience, when the distribution is centered at the origin ($\boldsymbol{c} = \mathbf{0}$), we denote it by $\mathfrak{D}_{\Lambda,\psi}$.

Definition 25 (Learning with errors (LWE)) *[11] Let* q *be a prime modulus,* $\alpha \in (0, 1)$ *a parameter and* $\boldsymbol{s} \in \mathbb{Z}_q^n$ *a secret vector sampled uniformly at random. We define two oracles as follows:*

1. **Setup**: The challenger $\mathcal{C}$ executes the $\mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Setup}(1^\lambda)$ to produce the master key pair $(\mathsf{Mpk}, \mathsf{Msk})$ and sends Mpk to the adversary $\mathcal{A}$. It maintains a decryption key query list $\mathsf{d}_{\mathsf{List}}$ (initially empty).
2. **Phase I**: The adversary $\mathcal{A}$ may issue a polynomial number of decryption key queries to the key extraction oracle $\mathcal{O}_{\mathsf{KE}}(\mathsf{Mpk}, \mathsf{Msk}, \cdot)$. For each query $\boldsymbol{y} \in \mathcal{Y}$, the challenger $\mathcal{C}$ responds by returning a decryption key $\mathsf{dk}_{\boldsymbol{y}} \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\ \mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y})$ to $\mathcal{A}$ and keeps a record of $\boldsymbol{y}$ in the $\mathsf{d}_{\mathsf{List}}$.
3. **Challenge**: The adversary $\mathcal{A}$ submits two challenge message vectors $\boldsymbol{x}^{(0)}$ and $\boldsymbol{x}^{(1)}$) to $\mathcal{C}$ where $\boldsymbol{x}^{(0)} = (x_i^{(0)})_{i \in I^*_{\boldsymbol{x}^{(0)}}}, \boldsymbol{x}^{(1)} = (x_i^{(1)})_{i \in I^*_{\boldsymbol{x}^{(1)}}} \in \mathcal{X}$ with $|I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}| = \ell$ and
   (a) either $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$ satisfying $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(j)} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(j)} \rangle$, for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$ whenever $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}} = I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$.
   (b) or $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$, for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$ whenever $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(1)}}$.

   The challenger $\mathcal{C}$ selects $\beta \xleftarrow{\$} \{0,1\}$, generates the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}}$ by running the algorithm $\mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}^{(\beta)})$ and provides $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}}$ to $\mathcal{A}$.
4. **Phase II**: The adversary $\mathcal{A}$ continues decryption key queries to $\mathcal{O}_{\mathsf{KE}}(\mathsf{Mpk}, \mathsf{Msk}, \cdot)$ as follows.
   (a) The adversary $\mathcal{A}$ sends secret key vectors $\boldsymbol{y}^{(j)}$ to $\mathcal{C}$ as in **Phase I** with the restriction that $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(j)} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(j)} \rangle$ for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}}$ with $\mathcal{R}_p(\boldsymbol{x}^{(\beta)}, \boldsymbol{y}^{(j)}) = 1$ i.e. $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}} = I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$.
   (b) The adversary $\mathcal{A}$ sends secret key vectors $\boldsymbol{y}^{(j)}$ to $\mathcal{C}$ with $\mathcal{R}_p(\boldsymbol{x}^{(\beta)}, \boldsymbol{y}^{(j)}) = 0$ i.e. $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(1)}}$ when $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$.

   The challenger $\mathcal{C}$ returns decryption key $\mathsf{dk}_{\boldsymbol{y}^{(j)}}$ to $\mathcal{A}$ and update the $\mathsf{d}_{\mathsf{List}}$.
5. **Guess**: Finally $\mathcal{A}$ outputs a bit $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KE}}(\mathsf{Mpk}, \mathsf{Msk}, \cdot)}(\mathsf{Mpk}, \mathsf{ct}_{\boldsymbol{x}^{(\beta)}})$ and sends β' to $\mathcal{C}$. The experiment outputs 1 if $\beta = \beta'$ and 0 otherwise.

Fig. 1. The adaptive indistinguishable security game $\mathsf{Expt}^{\mathsf{uIPFE}_{\mathsf{permissive}}}_{\mathcal{A}, \mathsf{Adp\text{-}IND}}(\lambda)$

- ***LWE Oracle*** ($\mathcal{O}_{\boldsymbol{s}}$)***:*** *Given an invocation, this oracle samples* $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n$ *uniformly and an error term* $x \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}_q, \alpha q}$ *from a discrete Gaussian distribution. It outputs the pair* $(\boldsymbol{a}, b = \boldsymbol{a}^\top \boldsymbol{s} + x) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.
- ***Uniform Oracle*** ($\mathcal{O}_{\$}$)***:*** *This oracle outputs a pair* $(\boldsymbol{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *sampled uniformly at random.*

Let $\mathcal{O}$ *be an oracle that is either* $\mathcal{O}_{\boldsymbol{s}}$ *or* $\mathcal{O}_{\$}$ *for all invocations. The* $\mathsf{LWE}_{q,\alpha,n}$ *decision problem is to distinguish between* $\mathcal{O}_{\boldsymbol{s}}$ *and* $\mathcal{O}_{\$}$ *with non-negligible advantage.*

Lemma 21 *[11] Let* $q \geq 2$ *be a prime modulus and* $\alpha \in (0,1)$ *a parameter such that* $\alpha q > 2\sqrt{n}$*. If there exists a polynomial-time algorithm for the decisional* $\mathsf{LWE}_{q,\alpha,n}$ *problem, then there exists an efficient quantum algorithm that approximates the* SIVP *and* GapSVP *problems in the* ℓ_2 *norm within an approximation factor of* $\mathcal{O}(n/\alpha)$.

Lemma 22 *[8],[2]: For any prime* $q \geq 2$*, any* $m \geq 2n \log q$*, there exists the following two* PPT *algorithms:*

- $\mathsf{TrapdoorGen}(n, m, q) \rightarrow (\mathbf{A}, \mathbf{T})$*: Given the parameters* n, m *and* q*, this algorithm outputs a random matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *that is statistically close to*

uniform and a short basis $\mathbf{T} \in \mathbb{Z}^{m\times m}$ *for* $\Lambda_q^{\perp}(\mathbf{A})$ *satisfying* $\mathbf{AT} = \mathbf{0}$ *with* $\|\widetilde{\mathbf{T}}\| \leq L = m^{2.5}$.

– $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}) \rightarrow \mathbf{Z}$*: This algorithm on input a matrix* $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$*, a short basis* $\mathbf{T} \in \mathbb{Z}^{m\times m}$*, a randomly chosen matrix* $\mathbf{U} \in \mathbb{Z}^{n\times \ell}$ *and any* $\sigma \geq L \cdot \omega(\sqrt{\log n})$ *with* $L = m^{2.5}$ *produces a matrix* $\mathbf{Z} \in \mathfrak{D}_{\mathbb{Z}^{m\times \ell},\sigma}$ *which satisfies* $\mathbf{AZ} = \mathbf{U}$ *mod* q.

Furthermore, it is established that the two distributions mentioned below are statistically close.

$$\left\{ (\mathbf{A}, \mathbf{Z}, \mathbf{U}) \;\middle|\; \begin{array}{l} (\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapdoorGen}(n, m, q) \\ \mathbf{Z} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}) \\ \textit{where } \mathbf{U} \xleftarrow{\$} \mathbb{Z}^{n\times \ell} \end{array} \right\} \approx_s \left\{ (\mathbf{A}, \mathbf{Z}, \mathbf{AZ}) \;\middle|\; \begin{array}{l} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_\ell) \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m\times \ell},\sigma} \\ \|\boldsymbol{z}_i\| \leq \sigma\sqrt{m}\ \forall i \in [\ell] \text{ where } \boldsymbol{z}_i \in \mathbb{Z}^m \text{ denotes} \\ \text{the } i\text{-th column of } \mathbf{Z} \text{ by Lemma 23.} \end{array} \right\}$$

Lemma 23 *[10] Let* $\Lambda \subset \mathbb{R}^n$ *be an* n*-dimensional lattice and* $\epsilon \in (0, 1)$*. Given a center* $\boldsymbol{c} \in \mathsf{span}(\Lambda)$[1] *and width* $\psi \geq \eta_\epsilon(\Lambda)$*, where* $\eta_\epsilon(\Lambda)$ *is the smoothing parameter*[2] *of* Λ*, a sample* $\boldsymbol{x} \leftarrow \mathfrak{D}_{\Lambda,\psi,\boldsymbol{c}}$ *satisfies:*

$$\Pr\left[\|\boldsymbol{x} - \boldsymbol{c}\| > \psi\sqrt{n}\right] \leq \left(\frac{1+\epsilon}{1-\epsilon}\right) 2^{-n}.$$

3 Our permissive uIPFE

Our permissive uIPFE scheme $\mathsf{uIPFE}_{\mathsf{permissive}} = (\mathsf{Setup}, \mathsf{KeyExtract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ over a real inner product space $\Gamma = \mathbb{R}^{\ell_{\mathsf{max}}}$ over $F = \mathbb{R}$ with the space of message vectors $\mathcal{X} = \{0, 1, \ldots, X-1\}^{\ell_{\mathsf{max}}} \subseteq \mathbb{R}^{\ell_{\mathsf{max}}}$, the secret key vector space $\mathcal{Y} = \{0, 1, \ldots, Y-1\}^{\ell_{\mathsf{max}}} \subseteq \mathbb{R}^{\ell_{\mathsf{max}}}$ and the permissive relation $\mathcal{R}_p : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ satisfying $\mathcal{R}_p = \{(\boldsymbol{x} = (x_i)_{i\in I_{\boldsymbol{x}}}, \boldsymbol{y} = (y_i)_{i\in I_{\boldsymbol{y}}}) \in \mathcal{X} \times \mathcal{Y} | I_{\boldsymbol{y}} \subseteq I_{\boldsymbol{x}}\}$ works as follows. Here the dimensions of message vector and secret key vector are each less than ℓ_{max} for some integers $\ell_{\mathsf{max}} = \ell_{\mathsf{max}}(\lambda), X = X(\lambda), Y = Y(\lambda) \in \mathbb{Z}$, $\lambda \in \mathbb{N}$ and the inner product of any two vectors $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{y} \in \mathcal{Y}$ results to be in the space $\{0, 1, \ldots, K-1\}$ where $K = XY\ell_{\mathsf{max}} = \mathsf{poly}(\lambda)$ (i.e. $\langle \boldsymbol{x}, \boldsymbol{y} \rangle < K\ \forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}$).

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{Mpk}, \mathsf{Msk})$: A trusted authority takes as input a security parameter $\lambda \in \mathbb{N}$ and performs the following steps.
  (i) Sets the parameters $n = n(\lambda)$, $m = m(\lambda)$, $X = X(\lambda)$, $Y = Y(\lambda)$, $\ell_{\mathsf{max}} = \ell_{\mathsf{max}}(\lambda) \in \mathbb{Z}$, prime modulus $q = q(\lambda) \in \mathbb{Z}$, the discrete Gaussian parameters $\sigma = \sigma(\lambda)$, $\alpha = \alpha(\lambda)$ and $\psi = \psi(\lambda) \in \mathbb{R}$ satisfying $m \geq 2n \log q$, $\sigma \geq L\omega(\sqrt{\log n})$, $\psi \geq 2C_1\alpha q(\sqrt{m}+\sqrt{n}+\sqrt{\ell_{max}})$, $\psi \geq \omega(\sqrt{\log n})$ and computes $K = XY\ell_{\mathsf{max}} \in \mathbb{Z}$ where $L = m^{2.5}$ and C_1 is an integer.
  (ii) Selects a full-domain hash function $\mathcal{H} : \mathbb{N} \rightarrow \mathbb{Z}_q^n$, modeled as a random oracle.

[1] $\mathsf{span}(\Lambda) = \mathsf{span}(\mathbf{B}) = \{\mathbf{B}\boldsymbol{v} \mid \boldsymbol{v} \in \mathbb{R}^n\}$ where $\mathbf{B}$ is a basis of Λ

[2] The smoothing parameter $\eta_\epsilon(\Lambda)$ is the least value of s such that $\rho_{1/s}(\Lambda^* - \{\mathbf{0}\}) \leq \epsilon$ for an n-dimentional lattice Λ and positive real number $\epsilon > 0$ where $\rho_{1/s}$ is a Gaussian function centered at $\mathbf{0}$ with parameter $1/s > 0$ and $\Lambda^* = \{\boldsymbol{v} \in \mathsf{span}(\Lambda) \mid \forall \boldsymbol{u} \in \Lambda, \boldsymbol{u}^\top \boldsymbol{v} \in \mathbb{Z}\}$.

(iii) Runs the algorithm $\mathsf{TrapdoorGen}(n, m, q) \to (\mathbf{A}, \mathbf{T})$ as in Lemma 22 such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is statistically close to uniform and $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is a short basis for $\Lambda_q^{\perp}(\mathbf{A})$ satisfying $\mathbf{AT} = \mathbf{0}$ with $\|\widetilde{\mathbf{T}}\| \leq L = m^{2.5}$.

(iv) Sets the master public key $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$ which is made public and the master secret key $\mathsf{Msk} = \mathbf{T} \in \mathbb{Z}^{m \times m}$ which is kept secret.

- $\mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y}) \to \mathsf{dk}_{\boldsymbol{y}}$: This algorithm takes as input the $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$, $\mathsf{Msk} = \mathbf{T} \in \mathbb{Z}^{m \times m}$ and a secret key vector $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}} \in \mathcal{Y}$ for a user with $I_{\boldsymbol{y}} = \{i'_1, i'_2, \ldots, i'_{\ell'}\}$, $\ell' \leq \ell_{\mathsf{max}}$ and works as follows. This algorithm is run by a trusted authority.
  (i) Computes a matrix $\mathbf{U}' = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{\ell'}) \in \mathbb{Z}_q^{n \times \ell'}$ where $\boldsymbol{u}'_k = \mathcal{H}(i'_k) \in \mathbb{Z}_q^n$ for each $i'_k \in I_{\boldsymbol{y}}$ and $1 \leq k \leq \ell'$.
  (ii) Runs the algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}') \to \mathbf{Z}$ as in Lemma 22 where $\sigma \geq L\omega(\sqrt{\log n})$ with $L = m^{2.5}$ and $\mathbf{Z} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell'}, \sigma}$ which satisfies $\mathbf{AZ} = \mathbf{U}' (\mathrm{mod}\ q) \in \mathbb{Z}_q^{n \times \ell'}$.
  (iii) Sets the decryption key corresponding to $\boldsymbol{y} \in \mathcal{Y}$ as $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \boldsymbol{y}^\top \mathbf{Z}^\top)$ and sends $\mathsf{dk}_{\boldsymbol{y}}$ to the user.
- $\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}) \to \mathsf{ct}_{\boldsymbol{x}}$: An encrypter on input the master public key $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$ and a message vector $\boldsymbol{x} = (x_i)_{i \in I_{\boldsymbol{x}}} \in \mathcal{X}$ with $I_{\boldsymbol{x}} = \{i_1, i_2, \ldots, i_\ell\}$, $\ell \leq \ell_{\mathsf{max}}$, executes the steps below.
  (i) Samples column vectors $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m, \psi}$, $\boldsymbol{e}_2 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^\ell, \psi}$ satisfying $\|\boldsymbol{e}_1\| \leq \psi\sqrt{m}$, $\|\boldsymbol{e}_2\| \leq \psi\sqrt{\ell}$.
  (ii) Computes $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\ell) \in \mathbb{Z}_q^{n \times \ell}$ where $\boldsymbol{u}_k = \mathcal{H}(i_k) \in \mathbb{Z}_q^n$ for each $i_k \in I_{\boldsymbol{x}}$, $1 \leq k \leq \ell$.
  (iii) Generates the ciphertext $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ associated with the message vector $\boldsymbol{x} \in \mathcal{X}$ of the form $\boldsymbol{c}_1 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}_1 \in \mathbb{Z}_q^m, \boldsymbol{c}_2 = \mathbf{U}^\top \boldsymbol{s} + \boldsymbol{e}_2 + \lfloor \frac{q}{K} \rfloor \boldsymbol{x} \in \mathbb{Z}_q^\ell$. Note that $\boldsymbol{c}_2 \in \mathbb{Z}_q^\ell$ can be written as $\boldsymbol{c}_2 = (c_{2,1}, c_{2,2}, \ldots, c_{2,\ell})$ where $c_{2,k} = \boldsymbol{u}_k^\top \boldsymbol{s} + e_{2,k} + \lfloor \frac{q}{K} \rfloor x_{i_k}$ with $\boldsymbol{e}_2 = (e_{2,1}, e_{2,2}, \ldots, e_{2,\ell})$, $\boldsymbol{x} = (x_{i_1}, x_{i_2}, \ldots, x_{i_\ell})$ and $1 \leq k \leq \ell$.
- $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{dk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) \to \mu / \perp$: The decryptor takes the master public key $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$, its own decryption key $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \boldsymbol{y}^\top \mathbf{Z}^\top)$ with $\boldsymbol{y} = (y_{i'_1}, y_{i'_2}, \ldots, y_{i'_{\ell'}})$, $I_{\boldsymbol{y}} = \{i'_1, i'_2, \ldots, i'_{\ell'}\}$ and a ciphertext $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ with $\boldsymbol{c}_2 = (c_{2,1}, c_{2,2}, \ldots, c_{2,\ell})$ and proceeds as follows.
  (i) If $\mathcal{R}_p(\boldsymbol{x}, \boldsymbol{y}) = 1$ i.e. $I_{\boldsymbol{y}} \subseteq I_{\boldsymbol{x}}$, then the decryption proceeds as follows, otherwise outputs $\perp$.
  (ii) Calculates $\mu' = \sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} c_{2,k} - (\boldsymbol{y}^\top \mathbf{Z}^\top) \boldsymbol{c}_1$.
  (iii) Outputs μ by searching over the set $\{0, 1, \ldots, K-1\}$ for which the quantity $|\lfloor \frac{q}{K} \rfloor \mu - \mu'|$ is minimized.

Correctness : Note that $\mathbf{AZ} = \mathbf{U}'$ by the correctness of algorithm $\mathsf{SamplePre}$ in Lemma 22. Also $\mathcal{R}_p(\boldsymbol{x}, \boldsymbol{y}) = 1 \implies I_{\boldsymbol{y}} \subseteq I_{\boldsymbol{x}}$. Thus $\boldsymbol{u}_k = \mathcal{H}(i_k) = \mathcal{H}(i'_k) = \boldsymbol{u}'_k$ $\forall i_k \in I_{\boldsymbol{x}} \cap I_{\boldsymbol{y}} = I_{\boldsymbol{y}}$.
Consequently, $\sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} \boldsymbol{u}_k^\top = \sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} \boldsymbol{u}'^\top_k = \boldsymbol{y}^\top \mathbf{U}'^\top$.

$$\begin{aligned}
\text{Hence, } \mu' &= \sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} c_{2,k} - (\boldsymbol{y}^\top \mathbf{Z}^\top)\boldsymbol{c}_1 \\
&= \sum_{i'_k \in I_{\boldsymbol{y}}} y_{i'_k} (\boldsymbol{u}_k^\top \boldsymbol{s} + e_{2,k} + \left\lfloor \frac{q}{K} \right\rfloor x_{i_k}) - (\boldsymbol{y}^\top \mathbf{Z}^\top)(\mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}_1) \\
&= \boldsymbol{y}^\top \mathbf{U}'^\top \boldsymbol{s} + \boldsymbol{y}^\top \boldsymbol{e}_2 + \left\lfloor \frac{q}{K} \right\rfloor \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \boldsymbol{y}^\top \mathbf{U}'^\top \boldsymbol{s} - \boldsymbol{y}^\top \mathbf{Z}^\top \boldsymbol{e}_1 \\
&= \left\lfloor \frac{q}{K} \right\rfloor \langle \boldsymbol{x}, \boldsymbol{y} \rangle + \underbrace{\boldsymbol{y}^\top \boldsymbol{e}_2 - \boldsymbol{y}^\top \mathbf{Z}^\top \boldsymbol{e}_1}_{\mathsf{noise}}
\end{aligned}$$

By the algorithm $\mathsf{SamplePre}$ in lemma 22, every column of $\mathbf{Z} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell'}, \sigma}$ is bounded above by $\sigma\sqrt{m}$ yielding $\|\mathbf{Z}\| \leq \sigma\sqrt{m\ell'}$ and $\|\boldsymbol{y}\| \leq Y\sqrt{\ell'}$ as $\boldsymbol{y} \in \mathcal{Y}$ with $\ell' \leq \ell_{\mathsf{max}}$. Also $\boldsymbol{e}_1 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^m, \psi}$ and $\boldsymbol{e}_2 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^\ell, \psi}$ implies $\|\boldsymbol{e}_1\| \leq \psi\sqrt{m}$ and $\|\boldsymbol{e}_2\| \leq \psi\sqrt{\ell}$ with $\ell \leq \ell_{\mathsf{max}}$ as long as $\psi \geq \omega(\sqrt{\log n})$. Therefore $\|\boldsymbol{y}^\top \boldsymbol{e}_2\| \leq Y\psi\sqrt{\ell\ell'}$. As $\psi \geq 2C_1\alpha q(\sqrt{m} + \sqrt{n} + \sqrt{\ell_{\mathsf{max}}})$ for a constant C_1, we have $\|(\boldsymbol{y}^\top \mathbf{Z}^\top)\boldsymbol{e}_1\| \leq \sigma\psi m\ell' Y$. Therefore, $|\mathsf{noise}| \leq Y\psi\sqrt{\ell\ell'} + \sigma\psi m\ell' Y \leq Y\psi\ell_{\mathsf{max}} + \sigma\psi m\ell_{\mathsf{max}} Y$. To guarantee successful decryption, the system requires the magnitude of the underlying noise to be bounded by $\frac{q}{2K}$. This condition is satisfied by selecting a modulus q that satisfies the inequality $q \geq 2K(Y\psi\ell_{\mathsf{max}} + \sigma\psi m\ell_{\mathsf{max}} Y)$.

Theorem 31 *Our permissive unbounded inner product functional encryption scheme* $\mathsf{uIPFE}_{\mathsf{permissive}} = (\mathsf{Setup}, \mathsf{KeyExtract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* $\mathsf{Adp\text{-}IND}$ *secure as per Definition 24 in the random oracle model assuming the* LWE *(see Definition 25) assumption.*

Proof. We establish the adaptive security of our construction in the random oracle model (ROM), building upon the proof structure and techniques introduced in [2]. To simplify the analysis, we assume, without loss of generality that before an adversary $\mathcal{A}$ submits its j-th decryption key query for a vector $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}}$, it first queries the random oracle (RO) $\mathcal{H}$ for the associated finite index set $I_{\boldsymbol{y}^{(j)}} \subseteq \mathbb{N}$, where $|I_{\boldsymbol{y}^{(j)}}| \leq \ell_{\max}$. Any standard adversary $\mathcal{A}$ can be efficiently transformed into an adversary $\mathcal{A}'$ that adheres to this behavior. The security of $\mathsf{uIPFE}_{\mathsf{permissive}}$ is demonstrated through a sequence of hybrid games between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. This proof accounts for challenge message vectors $\boldsymbol{x}^{(0)}$ and $\boldsymbol{x}^{(1)}$ under both cases where the index sets are identical ($I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$) or distinct ($I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$).

Game 0: This is the original security experiment $\mathsf{Expt}^{\mathsf{uIPFE}_{\mathsf{permissive}}}_{\mathcal{A}, \mathsf{Adp\text{-}IND}}$ as defined in Figure 2.

Game 1: In this game, the challenger $\mathcal{C}$ predicts the indices of the RO queries that the adversary $\mathcal{A}$ will make for the challenge index sets $I^*_{\boldsymbol{x}^{(0)}}$ and $I^*_{\boldsymbol{x}^{(1)}}$, where $|I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}| = \ell$. Directly guessing the index sets would result in an exponential security loss; therefore, $\mathcal{C}$ instead guesses the indices of the queries to the oracle $\mathcal{H}$ that yield the matrices $\mathbf{U}_0, \mathbf{U}_1 \in \mathbb{Z}_q^{n \times \ell}$. This reduction introduces a security loss factor of $1/Q_{\mathcal{H}}^2$, where $Q_{\mathcal{H}}$ represents the total number of queries to

the random oracle. Depending on the relationship between $I^*_{\boldsymbol{x}^{(0)}}$ and $I^*_{\boldsymbol{x}^{(1)}}$, the challenger reduces the security to either the modified ALS-IPFE assumption (see Appendix A) or the standard LWE problem.
The remaining steps of the proof follow directly from Lemma 31 and Lemma 32.

• **Setup** : The challenger $\mathcal{C}$ executes the $\mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Setup}(1^\lambda)$ algorithm to produce the master public-secret key pair $(\mathsf{Mpk}, \mathsf{Msk})$. The challenger $\mathcal{C}$ provides the adversary $\mathcal{A}$ with the master public key Mpk, while retaining the master secret key Msk where $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$ and $\mathsf{Msk} = \mathbf{T}$. Here $n, m, q, X, Y \in \mathbb{Z}$, $\psi, \sigma, \alpha \in \mathbb{R}$ are the discrete Gaussian parameters and $K = XY\ell_{\mathsf{max}}$ is polynomially bounded in λ, ℓ_{max} is the upper limit for the dimensions of both message and the secret key vectors, $\mathcal{H} : \mathbb{N} \to \mathbb{Z}_q^n$ is a hash function, modeled as random oracle and $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapdoorGen}(n, m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is statistically close to uniform and $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is a short basis for $\Lambda_q^\perp(\mathbf{A})$ satisfying $\mathbf{AT} = \mathbf{0}$ with $\|\widetilde{\mathbf{T}}\| \leq L$. These parameters satisfy $m \geq 2n \log q$, $\psi \geq \omega(\sqrt{\log n})$, $\sigma \geq L\omega(\sqrt{\log n})$, $\alpha q \geq 2\sqrt{n}$, $\psi \geq 2C_1 \alpha q(\sqrt{m} + \sqrt{n} + \sqrt{\ell_{\mathsf{max}}})$ and $q \geq 2K(Y\ell_{max}\psi + \sigma\psi m Y \ell_{\mathsf{max}})$ where $L = m^{2.5}$ and C_1 is a constant. It keeps Msk secret to itself and maintains a decryption key query list $\mathsf{d}_{\mathsf{List}}$ (initially empty).
• **Phase I** : The adversary $\mathcal{A}$ adaptively requests decryption keys for secret keys of its choice. For a query on $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathcal{Y}$ with $|I_{\boldsymbol{y}^{(j)}}| = \ell'$, the challenger $\mathcal{C}$ generates the decryption key $\mathsf{dk}_{\boldsymbol{y}^{(j)}} \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\ \mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y}^{(j)})$, provides it to $\mathcal{A}$ and keeps a record of $\boldsymbol{y}^{(j)}$ in $\mathsf{d}_{\mathsf{list}}$ where $\mathsf{dk}_{\boldsymbol{y}^{(j)}} = (\boldsymbol{y}^{(j)}, (\boldsymbol{y}^{(j)})^\top \mathbf{Z}^\top)$. The matrix $\mathbf{Z} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell'}, \sigma}$ is generated by the algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}') \to \mathbf{Z}$ satisfies $\mathbf{AZ} = \mathbf{U}' (\mathrm{mod}\ q)$ where $\mathbf{U}' = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{\ell'}) \in \mathbb{Z}_q^{n \times \ell'}$ with k-th column $\boldsymbol{u}'_k = \mathcal{H}(i'_k) \in \mathbb{Z}_q^n$ for all $i'_k \in I_{\boldsymbol{y}^{(j)}}$.
• **Challenge** : The adversary $\mathcal{A}$ submits to $\mathcal{C}$ distinct challenge vectors $\boldsymbol{x}^{(0)} = (x_i^{(0)})_{i \in I^*_{\boldsymbol{x}^{(0)}}} \in \mathcal{X}$ and $\boldsymbol{x}^{(1)} = (x_i^{(1)})_{i \in I^*_{\boldsymbol{x}^{(1)}}} \in \mathcal{X}$ with $|I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}| = \ell$ and

(a) either $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$ satisfying $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(j)} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(j)} \rangle$, for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$ whenever $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}} = I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$.

(b) or $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$, for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$ whenever $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(1)}}$.

The challenger $\mathcal{C}$ chooses a bit $\beta \xleftarrow{\$} \{0, 1\}$ uniformly at random and sends the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \leftarrow \mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}^{(\beta)})$ to $\mathcal{A}$ where $\boldsymbol{c}_1 = \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}_1 \in \mathbb{Z}_q^m$, $\boldsymbol{c}_2 = \mathbf{U}^\top \boldsymbol{s} + \boldsymbol{e}_2 + \lfloor \frac{q}{K} \rfloor \boldsymbol{x}^{(\beta)} \in \mathbb{Z}_q^\ell$ with $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n, \boldsymbol{e}_1 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^m, \psi}, \boldsymbol{e}_2 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^\ell, \psi}, \mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\ell) \in \mathbb{Z}_q^{n \times \ell}$, $\boldsymbol{u}_k = \mathcal{H}(i_k) \in \mathbb{Z}_q^n$ for all $i_k \in I^*_{\boldsymbol{x}^{(\beta)}}$.
• **Phase II** : The adversary $\mathcal{A}$ makes polynomial number of decryption key queries as follows.

(a) The adversary $\mathcal{A}$ sends $\boldsymbol{y}^{(j)}$ to $\mathcal{C}$ as in **Phase I** with the restriction that $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(j)} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(j)} \rangle$ for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$ whenever $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}} = I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$.

(b) The adversary $\mathcal{A}$ sends secret key vectors $\boldsymbol{y}^{(j)}$ to $\mathcal{C}$ as in **Phase I** with $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(1)}}$ whenever $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$.

The challenger returns decryption key $\mathsf{dk}_{\boldsymbol{y}^{(j)}}$ to $\mathcal{A}$ exactly in the same way as in **Phase I**.
• **Guess** : Finally $\mathcal{A}$ guesses a bit $\beta' \in \{0, 1\}$ and sends to $\mathcal{C}$.

Fig. 2. The originial security game $\mathsf{Expt}^{\mathsf{uIPFE}_{\mathsf{permissive}}}_{\mathcal{A}, \mathsf{Adp\text{-}IND}}(\lambda)$

Lemma 31 *Our permissive uIPFE scheme* $\mathsf{uIPFE}_{\mathsf{permissive}}$ *= (Setup, KeyExtract, Encrypt, Decrypt) is Adp-IND secure as per Definition 24 in the random oracle model when* $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$ *for the two challenge message vectors* $\boldsymbol{x}^{(0)}$, $\boldsymbol{x}^{(1)}$ *submitted by the adversary in the challenge phase of the security experiment* $\mathsf{Expt}^{\mathsf{uIPFE}_{\mathsf{permissive}}}_{\mathcal{A}, \mathsf{Adp\text{-}IND}}$, *as the modified ALS-IPFE (see Appendix A) is Adp-IND secure under the LWE assumption.*

• **Setup** : Instead of generating the Mpk by invoking $\mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Setup}(1^\lambda)$, the challenger $\mathcal{C}$ uses $\mathsf{Mpk}_{\mathsf{ALS}}$ received from $\mathcal{B}$ and generates $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$ by setting $\ell_{\mathsf{max}} = \ell$ and selecting a hash function $\mathcal{H} : \mathbb{N} \to \mathbb{Z}_q^n$. Thus the matrix $\mathbf{A}$ in Game 1, is not generated from the algorithm $\mathsf{TrapdoorGen}$ as in Game 0. The challenger $\mathcal{C}$ forwards Mpk to $\mathcal{A}$ and maintains a list $\mathsf{d}_{\mathsf{List}}$ (initially empty) to keep record of decryption key query vectors.
• **Phase I** : The adversary $\mathcal{A}$ makes polynomial number of decryption key queries. For the j-th decryption key query on a secret key vector $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathcal{Y}$, $\mathcal{C}$ proceeds as follows.
−On query to the random oracle (RO) $\mathcal{H}$ for $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}}$ with $|I_{\boldsymbol{y}^{(j)}}| = \ell' \leq \ell = |I^*_{\boldsymbol{x}}|$, $\mathcal{C}$ chooses a matrix $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m \times \ell', \sigma}}$ and programs the RO $\mathcal{H}$ by setting $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \in \mathbb{Z}_q^{n \times \ell'}$ instead of generating $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{\ell'}) \in \mathbb{Z}_q^{n \times \ell'}$ with $\mathcal{H}(i'_k) = \boldsymbol{u}'_k \in \mathbb{Z}_q^n$, $\forall i'_k \in I_{\boldsymbol{y}^{(j)}}$ and finding $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}_{I_{\boldsymbol{y}^{(j)}}})$ as in Game 0. For the decryption key queries on $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathcal{Y}$ with $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}}$, the challenger $\mathcal{C}$ returns $\mathsf{dk}_{\boldsymbol{y}^{(j)}} = (\boldsymbol{y}^{(j)}, (\boldsymbol{y}^{(j)})^\top \mathbf{Z}^\top_{I_{\boldsymbol{y}^{(j)}}})$.
−On query to the RO $\mathcal{H}$ for $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}}$ with $|I_{\boldsymbol{y}^{(j)}}| = \ell' \leq \ell = |I^*_{\boldsymbol{x}}|$, $\mathcal{C}$ programs the RO $\mathcal{H}$ by setting $\mathbf{U}_{I^*_{\boldsymbol{x}}} = \mathbf{D}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times \ell}$ instead of generating $\mathbf{U}_{I^*_{\boldsymbol{x}}} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\ell)$ with $\mathcal{H}(i_k) = \boldsymbol{u}_k \in \mathbb{Z}_q^n, \forall i_k \in I^*_{\boldsymbol{x}}$. Then $\mathcal{C}$ also constructs a submatrix $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} \in \mathbb{Z}_q^{n \times \ell'}$ of $\mathbf{U}_{I^*_{\boldsymbol{x}}}$ by removing the k-th column of $\mathbf{U}_{I^*_{\boldsymbol{x}}}$ for all $i_k \in I^*_{\boldsymbol{x}} \backslash I_{\boldsymbol{y}^{(j)}}$. Thus $\mathbf{U}_{I_{\boldsymbol{y}}} \in \mathbb{Z}_q^{n \times \ell'}$ is generated from $\mathbf{U}_{I_{\boldsymbol{x}^*}} = \mathbf{D}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times \ell}$ in Game 1 instead of generating $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \left(\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{\ell'}\right)$ with $\mathcal{H}(i'_k) = \boldsymbol{u}'_k \in \mathbb{Z}_q^n, \forall i'_k \in I_{\boldsymbol{y}^{(j)}}$ and finding $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}_{I_{\boldsymbol{y}^{(j)}}})$ satisfying $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}$ as in Game 0. For the decryption key query on $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathcal{Y}$ with $I_{\boldsymbol{y}^{(j)}} \subseteq I_{\boldsymbol{x}^*}$ received from $\mathcal{A}$, $\mathcal{C}$ constructs $\hat{\boldsymbol{y}}^{(j)} = (\hat{y}_i^{(j)})_{i \in I^*_{\boldsymbol{x}}}$ as $\hat{y}_i^{(j)} = \begin{cases} y_i^{(j)} & \forall i \in I_{\boldsymbol{y}^{(j)}} \\ 0 & \forall i \in I^*_{\boldsymbol{x}} \setminus I_{\boldsymbol{y}^{(j)}} \end{cases}$ and forwards $\hat{\boldsymbol{y}}^{(j)}$ to the modified $\mathsf{ALS\text{-}IPFE}$ challenger $\mathcal{B}$ who in turn generates the decryption key $\mathsf{dk}_{\hat{\boldsymbol{y}}^{(j)}} = (\hat{\boldsymbol{y}}^{(j)}, (\hat{\boldsymbol{y}}^{(j)})^\top \mathbf{Z}^\top_{\mathsf{ALS}})$. Here $\mathbf{Z}_{\mathsf{ALS}} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell}, \sigma}$ is the master secret key of $\mathcal{B}$ satisfying $\mathbf{D}_{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}_{\mathsf{ALS}}$. The challenger $\mathcal{B}$ sends $\mathsf{dk}_{\hat{\boldsymbol{y}}^{(j)}}$ to $\mathcal{C}$ who forwards $\mathsf{dk}_{\boldsymbol{y}^{(j)}} = (\boldsymbol{y}^{(j)}, (\hat{\boldsymbol{y}}^{(j)})^\top \mathbf{Z}^\top_{\mathsf{ALS}})$ to $\mathcal{A}$. Observe that $(\hat{\boldsymbol{y}}^{(j)})^\top \mathbf{Z}^\top_{\mathsf{ALS}} = (\boldsymbol{y}^{(j)})^\top \mathbf{Z}^\top_{I_{\boldsymbol{y}^{(j)}}}$ where $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell', \sigma}}$ is set by removing k-th column of $\mathbf{Z}_{\mathsf{ALS}}$ for all $i_k \in I^*_{\boldsymbol{x}} \setminus I_{\boldsymbol{y}^{(j)}}$. The challenger returns the decryption key to $\mathcal{A}$ and also keeps a record of $\boldsymbol{y}^{(j)}$ in $\mathsf{d}_{\mathsf{List}}$.
• **Challenge** : The adversary $\mathcal{A}$ submits the challenge message vector pair $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)})$ to $\mathcal{C}$ with the restriction that $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(j)} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(j)} \rangle$ whenever $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}} = I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$ for all $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$. The challenger $\mathcal{C}$ in turn forwards it to $\mathcal{B}$. On receiving the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ from $\mathcal{B}$, the challenger $\mathcal{C}$ hands it to $\mathcal{A}$. Here $\boldsymbol{c}_1 = \mathbf{A}^\top_{\mathsf{ALS}}\boldsymbol{s} + \boldsymbol{e}_1 \in \mathbb{Z}_q^m, \boldsymbol{c}_2 = \mathbf{D}^\top_{\mathsf{ALS}}\boldsymbol{s} + \boldsymbol{e}_2 + \lfloor \frac{q}{K} \rfloor \boldsymbol{x}^{(\beta)} \in \mathbb{Z}_q^\ell$ with $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_1 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^m, \psi}$ and $\boldsymbol{e}_2 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^\ell, \psi}$.
• **Phase II** : The adversary $\mathcal{A}$ makes polynomial number of decryption key queries as in **Phase I** with the restriction that $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(j)} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(j)} \rangle \forall \boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathsf{d}_{\mathsf{List}}$ whenever $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}} = I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$ which are simulated exactly in the same way as in **Phase I**.
• **Guess** : The adversary $\mathcal{A}$ guesses a bit $\beta' \in \{0, 1\}$ and sends to $\mathcal{C}$.

Fig. 3. The simulated game of $\mathsf{uIPFE}_{\mathsf{permissive}}$ challenger using the modified $\mathsf{ALS\text{-}IPFE}$ reduction

Proof. The security of our $\mathsf{uIPFE}_{\mathsf{permissive}}$ for two challenge message vectors $\boldsymbol{x}^{(0)}$, $\boldsymbol{x}^{(1)}$ with $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$ is shown by playing two hybrid games between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Let $\mathsf{Adv}_{\mathcal{A},\mathrm{Game}_i}(\lambda)$ denotes the advantage of $\mathcal{A}$ in Game i.
Game 0: This is the original security experiment $\mathsf{Expt}^{\mathsf{uIPFE}_{\mathsf{permissive}}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}$ described in Figure 2 along with the challenge query with the given restriction $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}}$.
Game 1: The challenger $\mathcal{C}$ of $\mathsf{uIPFE}_{\mathsf{permissive}}$ uses the security of the modified $\mathsf{ALS\text{-}IPFE}$, recalled in Appendix A to demonstrate the indistinguishability of the

challenge ciphertext encrypting either $\boldsymbol{x}^{(0)}$ or $\boldsymbol{x}^{(1)}$ where the message vector pair $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)})$ is submitted by $\mathcal{A}$ to $\mathcal{C}$. Then $\mathcal{C}$ is considered as an intermediate adversary engaged in interaction with the modified $\mathsf{ALS\text{-}IPFE}$ challenger $\mathcal{B}$. The challenger $\mathcal{B}$ generates the public key $\mathsf{Mpk}_{\mathsf{ALS}} = (\mathsf{pp}, \mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$ of the modified $\mathsf{ALS\text{-}IPFE}$ and sends it to $\mathcal{A}$ where $\mathsf{pp} = (n, m, q, \ell, X, Y, K, \sigma, \alpha, \psi)$, $\mathbf{A}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{D}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times \ell}$ satisfying $\mathbf{D}_{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}_{\mathsf{ALS}}$ the parameters $n, m, q, X, Y, \sigma, \alpha, \psi$ are same as those in our $\mathsf{uIPFE}_{\mathsf{permissive}}$ described in Game 0 satisfying the same conditions, $\ell = |I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}|$, $K = XY\ell$ and $\mathbf{Z}_{\mathsf{ALS}} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell}, \sigma}$ is kept secret to $\mathcal{B}$. Using the knowledge of the index sets $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}} = I^*_{\boldsymbol{x}}$, the challenger $\mathcal{C}$ simulates the view of $\mathcal{A}$ as detailed in Figure 3.

To show that Game 0 described in Figure 2 and Game 1 are statistically indistinguishable, we analyze two cases: either $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}}$ or $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}}$. Let $\mathsf{Adv}_{\mathcal{A},\mathrm{Game}_i}(\lambda)$ denote the advantage of $\mathcal{A}$ in Game i. Clearly, the ciphertext distributions in Game 0 and Game 1 are identical. We now show that the distributions of the master public key and the decryption keys in the two games are also identical.

• For any $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}}$ such that $|I_{\boldsymbol{y}}| = \ell' \leq \ell = |I^*_{\boldsymbol{x}}|$, we note that the distribution $\mathsf{Dist}_0 = (\mathbf{A}, \mathbf{Z}_{I_{\boldsymbol{y}}}, \mathbf{U}_{I_{\boldsymbol{y}^{(j)}}})$ where $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapdoorGen}\ (n, m, q)$, $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{\ell'}) \in \mathbb{Z}_q^{n \times \ell'}$ with $\mathcal{H}(i'_k) = \boldsymbol{u}'_k \in \mathbb{Z}_q^n$ for all $i'_k \in I_{\boldsymbol{y}^{(j)}}$ and $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}_{I_{\boldsymbol{y}^{(j)}}})$ results in an identical view for the adversary as in Game 0 satisfying $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}$. Next, consider the distribution $\mathsf{Dist}_1 = (\mathbf{A}, \mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}, \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}})$ where $\mathbf{A} = \mathbf{A}_{\mathsf{ALS}} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m \times \ell'}, \sigma}$ results in an identical view for the adversary as in Game 1. By Lemma 22, the distributions Dist_0 and Dist_1 are statistically close and hence

$$|\mathsf{Adv}_{\mathcal{A},\mathrm{Game}_1(\lambda)} - \mathsf{Adv}_{\mathcal{A},\mathrm{Game}_0}(\lambda)| \leq \mathsf{Adv}^{\mathsf{ALS\text{-}IPFE}}_{\mathcal{C},\mathsf{Adp\text{-}IND}}(\lambda).$$

• For any $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}}$, we compare the distribution of the decryption keys $\mathsf{dk}_{\boldsymbol{y}^{(j)}}$. The decryption key $\mathsf{dk}_{\boldsymbol{y}^{(j)}} = (\boldsymbol{y}^{(j)}, (\boldsymbol{y}^{(j)})^\top \mathbf{Z}^\top_{I_{\boldsymbol{y}^{(j)}}})$ where $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapdoorGen}\ (n, m, q)$, $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_{\ell'}) \in \mathbb{Z}_q^{n \times \ell'}$ with $\mathcal{H}(i'_k) = \boldsymbol{u}'_k \in \mathbb{Z}_q^n$ for all $i'_k \in I_{\boldsymbol{y}}$ and $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \sigma, \mathbf{U}_{I_{\boldsymbol{y}^{(j)}}})$ satisfying $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}$ which follows an identical distribution to the one observed in Game 0. In contrast, the decryption key $\mathsf{dk}_{\boldsymbol{y}^{(j)}} = (\boldsymbol{y}^{(j)}, (\boldsymbol{y}'^{(j)})^\top \mathbf{Z}^\top_{\mathsf{ALS}}) = (\boldsymbol{y}^{(j)}, (\boldsymbol{y}^{(j)})^\top \mathbf{Z}^\top_{\boldsymbol{y}})$ where $\mathbf{D}_{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}_{\mathsf{ALS}}$ with $\mathbf{D}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times \ell}$, $\mathbf{A}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{Z}_{\mathsf{ALS}} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell}, \sigma}$ exhibits a distribution statistically indistinguishable from that in Game 1. Since the challenger $\mathcal{C}$ sets $\mathbf{A} = \mathbf{A}_{\mathsf{ALS}}$, $\mathbf{U}_{I^*_{\boldsymbol{x}}} = \mathbf{D}_{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}_{\mathsf{ALS}}$ implies $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}$ for $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}}$ in Game 1 by implicitly setting $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{Z}'_{\mathsf{ALS}}$ where $\mathbf{Z}'_{\mathsf{ALS}}$ is constructed by removing k-th column of $\mathbf{Z}_{\mathsf{ALS}}$ for all $i_k \in I^*_{\boldsymbol{x}} \setminus I_{\boldsymbol{y}^{(j)}}$, the distribution of the decryption keys in Game 0 and Game 1 are identical.

In the modified $\mathsf{ALS\text{-}IPFE}$ reduction, the matrix $\mathbf{U}_{I^*_{\boldsymbol{x}}} = \mathbf{D}_{\mathsf{ALS}}$ is no longer chosen uniformly at random, as it was in Game 0. Instead, it is sampled as $\mathbf{U}_{I^*_{\boldsymbol{x}}} = \mathbf{D}_{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}_{\mathsf{ALS}}$, where $\mathbf{Z}_{\mathsf{ALS}} \leftarrow \mathfrak{D}_{\mathbb{Z}^{m \times \ell}, \sigma}$. The challenger $\mathcal{C}$ programs the random oracle $\mathcal{H}$ such that the resulting matrix $\mathbf{U}_{I^*_{\boldsymbol{x}}}$ satisfies the structure of $\mathbf{D}_{\mathsf{ALS}}$. Consequently, $\mathbf{U}_{I^*_{\boldsymbol{x}}}$ is not statistically close to the uniform distribution

and the derived matrix $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}}$ also deviates from uniform. Nevertheless, this discrepancy does not invalidate the security proof, as the distribution of $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}}$ remains computationally indistinguishable from uniform, given its representation as $\mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}_{\mathsf{ALS}}\mathbf{Z}'_{\mathsf{ALS}}$ with $\mathbf{A}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}$ being implicitly set as $\mathbf{Z}'_{\mathsf{ALS}}$ by deleting $(\ell - \ell')$ columns of $\mathbf{Z}_{\mathsf{ALS}} \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m\times\ell},\sigma}$ corresponding to index $i_k \in I^*_{\boldsymbol{x}} \setminus I_{\boldsymbol{y}^{(j)}}$. Therefore, applying the Lemma 22 we can conclude that $\mathsf{Dist}_0 = (\mathbf{A}, \mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}, \mathbf{U}_{I_{\boldsymbol{y}^{(j)}}})$ with $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and uniform $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} \in \mathbb{Z}_q^{n\times\ell'}$ is statistically close to $\mathsf{Dist}_1 = (\mathbf{A}, \mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}, \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}})$ and

$$|\mathsf{Adv}_{\mathcal{A},\mathrm{Game}_1}(\lambda) - \mathsf{Adv}_{\mathcal{A},\mathrm{Game}_0}(\lambda)| \leq \mathsf{Adv}^{\mathsf{ALS\text{-}IPFE}}_{\mathcal{C},\mathsf{Adp\text{-}IND}}(\lambda).$$

Lemma 32 *Our permissive uIPFE scheme* $\mathsf{uIPFE}_{\mathsf{permissive}} = (\mathsf{Setup}, \mathsf{KeyExtract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is Adp-IND secure as per Definition 24 in the random oracle model when* $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$ *for the two challenge message vectors with* $|I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}|$ *assuming the hardness of LWE problem (see Definition 25).*

Proof. The security of our $\mathsf{uIPFE}_{\mathsf{permissive}}$ for two challenge message vectors $\boldsymbol{x}^{(0)}$, $\boldsymbol{x}^{(1)}$ with $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$ is shown by playing two hybrid games between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Let $\mathsf{Adv}_{\mathcal{A},\mathrm{Game}_i}(\lambda)$ represents the advantage of $\mathcal{A}$ in Game i.

Game 0: This is the original game presented by the experiment $\mathsf{Expt}^{\mathsf{uIPFE}_{\mathsf{permissive}}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}$ in Definition 24 and is detailed in Figure 2 along with the challenge query with the given restriction $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$.

Game 1: Using the same logic as Game 1 of the Theorem 31, we can say that if the adversary $\mathcal{A}$ does total $Q_{\mathcal{H}}$ number of random oracle (RO) queries to $\mathcal{H}$ for each index set query, then it results $\frac{1}{Q^2_{\mathcal{H}}}$ security loss. Here, the $\mathsf{uIPFE}_{\mathsf{permissive}}$ challenger $\mathcal{C}$ acts as an intermediate adversary interacting with an LWE challenger $\mathcal{B}$. Given a large prime $q \geq 2$, a positive integer $n \in \mathbb{N}$ and $\alpha \in (0,1) \cap \mathbb{R}$ such that $\alpha q > 2\sqrt{n}$ (see Lemma 21), the LWE challenger $\mathcal{B}$ outputs a LWE instance $\mathsf{LWE}_{\mathsf{inst}} = (\mathbf{E}, \boldsymbol{z}) \in \mathbb{Z}_q^{n\times(m+2\ell)} \times \mathbb{Z}_q^{m+2\ell}$ by either computing $\boldsymbol{z} = \mathbf{E}^\top \boldsymbol{g} + \boldsymbol{d}$ or sampling $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_q^{m+2\ell}$ uniformly at random and sends $\mathsf{LWE}_{\mathsf{inst}}$ to $\mathcal{C}$ where $\boldsymbol{z} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(0)}, \boldsymbol{c}_2^{(1)}) \in \mathbb{Z}_q^{m+2\ell}$, $\boldsymbol{g} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{E} = [\mathbf{A}|\mathbf{U}_0|\mathbf{U}_1] \xleftarrow{\$} \mathbb{Z}_q^{n\times(m+2\ell)}$ and $\boldsymbol{d} = (\boldsymbol{d}_1, \boldsymbol{d}_2^{(0)}, \boldsymbol{d}_2^{(1)}) \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m+2\ell},\psi}$ with $m \geq 2n\log q$, $\ell = |I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}|$ and $\psi = \alpha q$. Given $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$, the challenger $\mathcal{C}$ simulates the game for $\mathcal{A}$ as shown in Figure 4. Clearly, the distribution of the master public key and decryption keys in Game 0 and Game 1 are identical. Now we proceed for the distribution of ciphertexts.

• **Case 1 (Uniform):** When the LWE challenger $\mathcal{B}$ outputs a random pair $(\mathbf{E}, \boldsymbol{z})$ where $\mathbf{E} = [\mathbf{A}|\mathbf{U}_0|\mathbf{U}_1] \xleftarrow{\$} \mathbb{Z}_q^{n\times(m+2\ell)}$ and $\boldsymbol{z} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(0)}, \boldsymbol{c}_2^{(1)}) \xleftarrow{\$} \mathbb{Z}_q^{m+2\ell}$, the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(\beta)} + \lfloor \frac{q}{K} \rfloor \boldsymbol{x}^{(\beta)}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$ is distributed identically to the ciphertext generated in Game 1.

• **Case 2 (Valid LWE):** When the LWE challenger $\mathcal{B}$ outputs a pair $(\mathbf{E}, \boldsymbol{z})$ such that $\boldsymbol{z} = \mathbf{E}^\top \boldsymbol{g} + \boldsymbol{d}$, we have $\boldsymbol{z} = (\mathbf{A}^\top \boldsymbol{g} + \boldsymbol{d}_1, \mathbf{U}_0^\top \boldsymbol{g} + \boldsymbol{d}_2^{(0)}, \mathbf{U}_1^\top \boldsymbol{g} + \boldsymbol{d}_2^{(1)})$. In

this scenario, the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(\beta)} + \lfloor \frac{q}{K} \rfloor \boldsymbol{x}^{(\beta)})$ is distributed identically to the ciphertext generated in Game 0.
Thus, we bound the advantage difference as $|\mathsf{Adv}_{\mathcal{A},\text{Game }1}(\lambda) - \mathsf{Adv}_{\mathcal{A},\text{Game }0}(\lambda)| \leq \mathsf{Adv}_{\mathcal{C}}^{\mathsf{LWE}_{q,\alpha,n}}(\lambda)$, yielding:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Adp\text{-}IND}}(\lambda) \leq \frac{1}{Q_{\mathcal{H}}^2} \cdot \mathsf{Adv}_{\mathcal{C}}^{\mathsf{LWE}_{q,\alpha,n}}(\lambda)$$

where $Q_{\mathcal{H}}$ is the total number of random oracle queries.

• **Setup**: Instead of generating the Mpk by invoking $\mathsf{uIPFE}_{\mathsf{permissive}}.\mathsf{Setup}(1^\lambda)$, the challenger $\mathcal{C}$ uses the $\mathsf{LWE}_{\mathsf{inst}}$ to derive $\mathbf{A}$ and generates $\mathsf{Mpk} = (n, m, q, \ell_{\mathsf{max}}, X, Y, K, \sigma, \alpha, \psi, \mathcal{H}, \mathbf{A})$ by choosing a hash function $\mathcal{H} : \mathbb{N} \to \mathbb{Z}_q^n$ and selecting the parameters $\ell_{\mathsf{max}}, X, Y, K = XY\ell_{\mathsf{max}} \in \mathbb{Z}$, $\sigma, \psi = \alpha q \in \mathbb{R}$ satisfying the same conditions as Game 0. Consequently, the matrix $\mathbf{A}$ in Game 1 is not generated using the algorithm $\mathsf{TrapdoorGen}$ as in Game 0. The challenger $\mathcal{C}$ forwards Mpk to $\mathcal{A}$ and maintains a list $\mathsf{d}_{\mathsf{List}}$ (initiialy empty) to keep record of decryption key query vectors.
• **Phase I**: According to the $\mathsf{Adp\text{-}IND}$ security model (see Definition 24), when $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$, the adversary $\mathcal{A}$ is not allowed to obtain the decryption key for $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}} \in \mathcal{Y}$ with $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}^{(0)}}$ or $I_{\boldsymbol{y}^{(j)}} \subseteq I^*_{\boldsymbol{x}^{(1)}}$ or $I_{\boldsymbol{y}^{(j)}} \subseteq (I^*_{\boldsymbol{x}^{(1)}} \cap I^*_{\boldsymbol{x}^{(1)}})$ as using these decryption keys, $\mathcal{A}$ can easily distinguish the encryption of $\boldsymbol{x}^{(0)} = (x_i^{(0)})_{i \in I^*_{\boldsymbol{x}^{(0)}}}$ and $\boldsymbol{x}^{(1)} = (x_i^{(1)})_{i \in I^*_{\boldsymbol{x}^{(1)}}}$. Thus the adversary $\mathcal{A}$ can query for decryption keys corresponding to the index sets $I_{\boldsymbol{y}^{(j)}}$ with $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(1)}}$ whenever $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$. To simulate the view of $\mathcal{A}$, the challenger sets $\mathcal{H}(i_k) = \boldsymbol{u}_k \in \mathbb{Z}_q^n$ for all $i_k \in I^*_{\boldsymbol{x}^{(0)}}$ and $\mathcal{H}(i'_k) = \boldsymbol{u}'_k \in \mathbb{Z}_q^n$ for all $i'_k \in I^*_{\boldsymbol{x}^{(1)}}$ where $\mathbf{U}_0 = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\ell) \in \mathbb{Z}_q^{n \times \ell}$ and $\mathbf{U}_1 = (\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_\ell) \in \mathbb{Z}_q^{n \times \ell}$ are derived from $\mathsf{LWE}_{\mathsf{inst}}$. For any query to the random oracle (RO) $\mathcal{H}$ for $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(1)}}$ with $|I_{\boldsymbol{y}^{(j)}}| = \ell' \leq \ell = |I^*_{\boldsymbol{x}^{(0)}}| = |I^*_{\boldsymbol{x}^{(1)}}|$, the challenger chooses a matrix $\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}} \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m \times \ell'}, \sigma}$ with $|I_{\boldsymbol{y}^{(j)}}| = \ell'$ and programs the RO $\mathcal{H}$ such that the outputs $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}}$ is computed as $\mathbf{U}_{I_{\boldsymbol{y}^{(j)}}} = \mathbf{A}\mathbf{Z}_{I_{\boldsymbol{y}^{(j)}}}$. For each query of decryption key corresponding to $\boldsymbol{y}^{(j)} = (y_i^{(j)})_{i \in I_{\boldsymbol{y}^{(j)}}}$, the adversary gets the decryption key $\mathsf{dk}_{\boldsymbol{y}^{(j)}} = (\boldsymbol{y}^{(j)}, (\boldsymbol{y}^{(j)})^\top \mathbf{Z}^\top_{I_{\boldsymbol{y}^{(j)}}})$. The challenger keeps a record of $\boldsymbol{y}^{(j)}$ in $\mathsf{d}_{\mathsf{List}}$.

• **Challenge**: The adversary submits the challenge message vector pair $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)})$ to $\mathcal{C}$ whenever $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ with $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$ for all $\boldsymbol{y}^{(j)} \in \mathsf{d}_{\mathsf{List}}$. Now $\mathcal{C}$ chooses a random bit $\beta \xleftarrow{\$} \{0,1\}$ to answer challenge query for the message vector $\boldsymbol{x}^{(\beta)}$. For $\beta = 0$, the challenger returns $\mathsf{ct}_{\boldsymbol{x}^{(0)}} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(0)} + \lfloor \frac{q}{K} \rfloor \boldsymbol{x}^{(0)})$ and for $\beta = 1$, it returns $\mathsf{ct}_{\boldsymbol{x}^{(1)}} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(1)} + \lfloor \frac{q}{K} \rfloor \boldsymbol{x}^{(1)})$ where $\boldsymbol{c}_1$, $\boldsymbol{c}_2^{(0)}$ and $\boldsymbol{c}_2^{(1)}$ are obtained by spliting $\boldsymbol{z} = (\boldsymbol{c}_1, \boldsymbol{c}_2^{(0)}, \boldsymbol{c}_2^{(1)})$ derived from $\mathsf{LWE}_{\mathsf{inst}}$.
• **Phase II**: The adversary $\mathcal{A}$ makes polynomial number of decryption key queries as in **Phase I** whenever $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ and $I_{\boldsymbol{y}^{(j)}} \nsubseteq I^*_{\boldsymbol{x}^{(0)}}$ with $I^*_{\boldsymbol{x}^{(0)}} \neq I^*_{\boldsymbol{x}^{(1)}}$ for all $\boldsymbol{y}^{(j)} \in \mathsf{d}_{\mathsf{List}}$ which are simluated exactly in the same way as in **Phase I**.
• **Guess**: The adversary $\mathcal{A}$ guesses a bit $\beta' \in \{0,1\}$ and sends to $\mathcal{C}$.

Fig. 4. The simulated game of $\mathsf{uIPFE}_{\mathsf{permissive}}$ challenger using the LWE reduction

4 Conclusion

This work presents the first permissive public-key unbounded inner product functional encryption scheme that is resilient against quantum threats. This construction ensures adaptive security, grounded in the LWE assumption, within

the ROM and is distinctive for not employing pairing-based techniques. However, its reliance on the ROM is identified as a limitation, highlighting the necessity for a lattice-based public-key permissive uIPFE scheme that guarantees provable security in the standard model. Additionally, achieving chosen-ciphertext security for uIPFE poses significant technical hurdles, such as managing unbounded vector dimensions, preventing leakage accumulation, addressing adaptively chosen queries and balancing efficiency concerns which are earmarked for future research.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: IACR International Workshop on Public Key Cryptography. pp. 733–751. Springer (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 467–497. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_16
3. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Annual International Cryptology Conference. pp. 333–362. Springer (2016). https://doi.org/10.1007/978-3-662-53015-3_12
4. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011). https://doi.org/10.1007/978-3-642-19571-6_16
5. Cini, V., Tairi, E.: (fine-grained) unbounded inner-product functional encryption from lwe. Cryptology ePrint Archive (2026)
6. Dufour-Sans, E., Pointcheval, D.: Unbounded inner-product functional encryption with succinct keys. In: International Conference on Applied Cryptography and Network Security. pp. 426–441. Springer (2019). https://doi.org/10.1007/978-3-030-21568-2_21
7. Dutta, S., Dutta, R., Mukhopadhyay, S.: Securing data in the cloud using pairing-free inner product functional encryption with unbounded vector size. Theoretical Computer Science **1031**, 115085 (2025). https://doi.org/10.1016/j.tcs.2025.115085
8. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206 (2008). https://doi.org/10.1145/1374376.1374407
9. Li, Y., Ng, K.S., Purcell, M.: A tutorial introduction to lattice-based cryptography and homomorphic encryption. arXiv preprint arXiv:2208.08125 (2022). https://doi.org/10.48550/arXiv.2208.08125
10. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM journal on computing **37**(1), 267–302 (2007). https://doi.org/10.1137/S0097539705447360
11. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography, 2005. In: STOC. pp. 84–93. ACM (2005). https://doi.org/10.1145/1568318.1568324

12. Tomida, J., Takashima, K.: Unbounded inner product functional encryption from bilinear maps. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II 24. pp. 609–639. Springer (2018). https://doi.org/10.1007/s13160-020-00419-x

A Modified ALS-IPFE [2]

The modified ALS-IPFE = (Setup, KeyExtract, Encrypt, Decrypt) works as follows. Unlike ALS-IPFE by Agrawal et al. [3], it allows selection of secret matrix $\mathbf{Z}$ from the discrete Gaussian distribution. Considering the space of message vectors $\mathcal{X} = \{0, 1, \ldots, X-1\}^\ell$, the space of secret key vectors $\mathcal{Y} = \{0, 1, \ldots, Y-1\}^\ell$ with the same length of message vectors and secret key vectors denoted by ℓ and the inner product value is assumed to be in the set $\{0, 1, \ldots, K-1\}$ where $K = \ell XY$, $X = X(\lambda), Y = Y(\lambda)$. In other words, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle < K$, $\forall \boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$.

– $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{Mpk}, \mathsf{Msk})$: On input a security parameter $\lambda \in \mathbb{N}$ and the length of input vector $\ell \in \mathbb{N}$, the trusted authority chooses the parameters as Table 4 and runs the setup algorithm as in Algorithm 1 to generate the master public key Mpk and master secret key Msk. The Mpk is published, Msk is kept secret.

Algorithm 1 $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{Mpk}, \mathsf{Msk})$

Input : λ; ℓ;
Output : $\mathsf{Mpk} = (\mathsf{pp}, \mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$; $\mathsf{pp} = (n, m, q, \ell, X, Y, K, \sigma, \alpha, \psi)$; $\mathsf{Msk} = \mathbf{Z}_{\mathsf{ALS}}$;
1: $n = n(\lambda)$, $m \geq 2n \log q$, $X = X(\lambda)$, $Y = Y(\lambda)$, $K = \ell XY$, a prime $q(\geq 2) \in \mathbb{N}$; $\psi \in \mathbb{R}$, $\alpha \in (0,1) \cap \mathbb{R}$, $\sigma \in \mathbb{R}$ satisfying the condition in Table 4;
2: $\mathbf{A}_{\mathsf{ALS}} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{Z}_{\mathsf{ALS}} \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^{m \times \ell}, \sigma}$ satisfying $\|\boldsymbol{z}_i\| \leq \sigma\sqrt{m}$ $\forall i \in [\ell]$ where $\boldsymbol{z}_i$ is the i-th column of $\mathbf{Z}_{\mathsf{ALS}}$;
3: $\mathbf{D}_{\mathsf{ALS}} = \mathbf{A}_{\mathsf{ALS}} \mathbf{Z}_{\mathsf{ALS}} \in \mathbb{Z}_q^{n \times \ell}$;
4: **return** $(n, m, q, \ell, X, Y, K, \sigma, \alpha, \psi, \mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}}, \mathbf{Z}_{\mathsf{ALS}})$;

– $\mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y}) \to \mathsf{dk}_{\boldsymbol{y}}$: The trusted authority intakes the master public key Mpk, master secret key Msk and secret key vector $\boldsymbol{y} \in \mathcal{Y}$ for a user, runs the following Algorithm 2 to generate a decryption key $\mathsf{dk}_{\boldsymbol{y}}$ and sends to the user through a secure communication channel between them.

Algorithm 2 $\mathsf{KeyExtract}(\mathsf{Mpk}, \mathsf{Msk}, \boldsymbol{y}) \to \mathsf{dk}_{\boldsymbol{y}}$

Input : $\mathsf{Mpk} = (\mathsf{pp}, \mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$; $\mathsf{Msk} = \mathbf{Z}_{\mathsf{ALS}} \in \mathfrak{D}_{\mathbb{Z}^{m \times \ell}, \sigma}$; $\boldsymbol{y} \in \mathcal{Y}$;
Output : $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{k}_1, \boldsymbol{k}_2)$;
1: $\boldsymbol{k}_1 = \boldsymbol{y} \in \mathcal{Y}$, $\boldsymbol{k}_2 = \mathbf{Z}_{\mathsf{ALS}} \boldsymbol{y} \in \mathbb{Z}_q^m$;
2: **return** $(\boldsymbol{k}_1, \boldsymbol{k}_2)$;

– $\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}) \rightarrow \mathsf{ct}_{\boldsymbol{x}}$: The encryptor runs the following Algorithm 3 on input the master public key Mpk and a message vector $\boldsymbol{x} \in \mathcal{X}$ to generate a ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ and sends over a public channel.

Algorithm 3 $\mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}) \rightarrow \mathsf{ct}_{\boldsymbol{x}}$

Input : $\mathsf{Mpk} = (\mathsf{pp}, \mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$; $\boldsymbol{x} \in \mathcal{X}$;
Output : $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$;
1: $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_1 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^m,\psi}$, $\boldsymbol{e}_2 \xleftarrow{\$} \mathfrak{D}_{\mathbb{Z}^\ell,\psi}$ satisfying $\|\boldsymbol{e}_1\| \leq \psi\sqrt{m}$, $\|\boldsymbol{e}_2\| \leq \psi\sqrt{\ell}$;
2: $\boldsymbol{c}_1 = \mathbf{A}_{\mathsf{ALS}}^\top \boldsymbol{s} + \boldsymbol{e}_1 \in \mathbb{Z}_q^m$;
3: $\boldsymbol{c}_2 = \mathbf{D}_{\mathsf{ALS}}^\top \boldsymbol{s} + \boldsymbol{e}_2 + \lfloor \frac{q}{K} \rfloor \boldsymbol{x} \in \mathbb{Z}_q^\ell$;
4: **return** $(\boldsymbol{c}_1, \boldsymbol{c}_2)$;

– $\mathsf{Decrypt}(\mathsf{Mpk}, \mathsf{dk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) \rightarrow \mu \vee \perp$: The decryptor executes the decryption algorithm as displayed in Algorithm 4 on input the master public key Mpk, its decryption key $\mathsf{dk}_{\boldsymbol{y}}$ and a ciphertext $\mathsf{ct}_{\boldsymbol{x}}$,.

Algorithm 4 $\mathsf{Decrypt}(\mathsf{Mpk}, \mathsf{dk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) \rightarrow \mu \vee \perp$

Input : $\mathsf{Mpk} = (\mathsf{pp}, \mathbf{A}_{\mathsf{ALS}}, \mathbf{D}_{\mathsf{ALS}})$; $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{k}_1, \boldsymbol{k}_2)$; $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$;
Output : $\mu \in \{0, 1\}$;
1: $\mu' = \langle \boldsymbol{k}_1, \boldsymbol{c}_2 \rangle - \langle \boldsymbol{k}_2, \boldsymbol{c}_1 \rangle \in \mathbb{Z}_q$;
2: **for** ($\mu = 0$ to $K - 1$) **do**
3: **if** $(|\lfloor \frac{q}{K} \rfloor \mu - \mu'| < (\sigma\psi m\ell Y + Y\ell\psi))$ **then**
4: **break**;
5: **end if**
6: **end for**
7: **return** μ;

Lemma A1 (Correctness) *[2] The modified* ALS-IPFE *is correct provided* $\psi, \sigma > \omega(\sqrt{\log n})$ *and* $q > 2\ell K\sqrt{\ell}Y\omega(\log^2 n)$.

Table 4. Parameters for the modified ALS-IPFE

Parameters	Constraints
m, n	$m \geq 2n \log q$
ψ, σ	$\psi \geq \omega(\sqrt{\log n}), \sigma \geq \omega(\sqrt{\log n})$
α, q, n	$\alpha q \geq 2\sqrt{n}$, $\alpha \in (0, 1) \cap \mathbb{R}$
α, q, m, n, ℓ	$\alpha \leq \frac{\psi}{2C_1 q(\sqrt{m}+\sqrt{n}+\sqrt{\ell_{max}})}$ where C_1 is a constant
$q, \sigma, \psi, \ell, X, Y$	$q \geq 2\ell K\sqrt{\ell}Y\omega(\log^2 n)$ where $K = \ell XY$

Definition A1 (Adaptive indistinguishability (Adp-IND)) *[2] The modified ALS-IPFE = (Setup, KeyExtract, Encrypt, Decrypt) is said to satisfy adaptive indistinguishable* (Adp-IND) *security if for every PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$, the following holds,*

$$\mathsf{Adv}^{\mathsf{ALS-IPFE}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda) = \left| \Pr\left[\mathsf{Expt}^{\mathsf{ALS-IPFE}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{ALS-IPFE}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda)$ *is defined in Figure 5.*

1. $(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KE}}(\mathsf{Mpk},\mathsf{Msk},\cdot)}(\mathsf{Mpk}) \mid \boldsymbol{x}^{(0)} = (x_i^{(0)})_{i \in I^*_{\boldsymbol{x}^{(0)}}}, \boldsymbol{x}^{(1)} = (x_i^{(1)})_{i \in I^*_{\boldsymbol{x}^{(1)}}}$ with $I^*_{\boldsymbol{x}^{(0)}} = I^*_{\boldsymbol{x}^{(1)}} = I^*_{\boldsymbol{x}}$.
3. $\beta \xleftarrow{\$} \{0,1\}$
4. $\mathsf{ct}_{\boldsymbol{x}^{(\beta)}} \leftarrow \mathsf{Encrypt}(\mathsf{Mpk}, \boldsymbol{x}^{(\beta)})$
5. $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KE}}(\mathsf{Mpk},\mathsf{Msk},\cdot)}(\mathsf{Mpk}, \mathsf{ct}_{\boldsymbol{x}^{(\beta)}})$
6. Output 1 if $\beta = \beta'$ else 0.

Fig. 5. $\mathsf{Expt}^{\mathsf{ALS-IPFE}}_{\mathcal{A},\mathsf{Adp\text{-}IND}}(\lambda)$

In the experiment, the oracle $\mathcal{O}_{\mathsf{KE}}(\mathsf{Mpk}, \mathit{Msk}, \cdot)$ takes input a secret key vector $\boldsymbol{y} = (y_i)_{i \in I_{\boldsymbol{y}}}$ satisfying $\langle \boldsymbol{x}^{(0)}, \boldsymbol{y} \rangle = \langle \boldsymbol{x}^{(1)}, \boldsymbol{y} \rangle$ whenever $I^*_{\boldsymbol{x}} = I_{\boldsymbol{y}}$ and outputs the decryption key $\mathsf{dk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyExtract}(\mathsf{Mpk}, \mathit{Msk}, \boldsymbol{y})$.

Lemma A2 (Security) *[2] The modified ALS-IPFE is Adp-IND (see Definition A1)secure under the hardness of the* LWE *problem (see Definition 25) provided the prime $q > 2\ell K\sqrt{\ell} Y \omega(\log^2 n)$, the integer $m \geq 2n \log q$, the real numbers $\psi \geq \omega(\sqrt{\log n}), \sigma \geq \omega(\sqrt{\log n})$ and $\alpha \leq \frac{\psi}{2C_1 q(\sqrt{m}+\sqrt{n}+\sqrt{\ell})}$ where C_1 is a constant.*