Compact Inner Product Encryption from LWE

Zhedong Wang[1], Xiong Fan[2], and Mingsheng Wang[1]

[1] State Key Laboratory of Information Security, Institute of Information Engineering,Chinese Academy of Sciences, Beijing, China.
{wangzhedong, wangmingsheng}@iie.ac.cn.*
[2] Cornell University, Ithaca, NY, USA. xfan@cs.cornell.edu.**

Abstract. Predicate encryption provides fine-grained access control and has attractive applications. In this paper, We construct an compact inner product encryption scheme from the standard Learning with Errors (LWE) assumption that has compact public-key and achieves weakly attribute-hiding in the standard model. In particular, our scheme only needs two public matrices to support inner product over vector space $\mathbb{Z}_q^{\log \lambda}$, and $(\lambda/\log \lambda)$ public matrices to support vector space $\mathbb{Z}_q^{\lambda}$.
Our construction is the first compact functional encryption scheme based on lattice that goes beyond the very recent optimizations of public parameters in identity-based encryption setting. The main technique in our compact IPE scheme is a novel combination of IPE scheme of Agrawal, Freeman and Vaikuntanathan (Asiacrypt 2011), fully homomorphic encryption of Gentry, Sahai and Waters (Crypto 2013) and vector encoding schemes of Apon, Fan and Liu (Eprint 2017).

1 Introduction

Encryption has traditionally been regarded as a way to ensure confidentiality of an end-to-end communication. However, with the emergence of complex networks and cloud computing, recently the crypto community has been re-thinking the notion of encryption to address security concerns that arise in these more complex environments. *Functional encryption* [11, 23], generalized from identity based encryption [25, 9] and attribute based encryption [19, 8], provides a satisfying solutions to this problem in theory. Two features provided by functional encryption are fine-grained access and computing on encrypted data. The fine-grained access part is formalized as a cryptographic notion, named *predicate encryption* [12, 20]. In predicate encryption system, each ciphertext ct is associated with an attribute a while each secret key sk is associated with a predicate f. A user holding the key sk can decrypt ciphertext ct if and only if $f(a) = 0$. Moreover, the attribute a is kept hidden.

* This work was supported by the National Science Foundation of China (No.61772516).
** This material is based upon work supported by IBM under Agreement 4915013672. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

With several significant improvements on quantum computing, the community is working intensively on developing applications whose security holds even against quantum attacks. Lattice-based cryptography, the most promising candidate against quantum attacks, has matured significantly since the early works of Ajtai [4] and Regev [24]. Most cryptographic primitives, ranging from basic public-key encryption (PKE) [24] to more advanced schemes e.g., identity-based encryption (IBE) [13, 2], attribute-based encryption (ABE) [18, 10], fully-homomorphic encryption (FHE) [15], etc., can be built from now canonical lattice hardness assumptions, such as Regev's Learning with Errors (LWE). From the above facts, we can draw the conclusion that our understanding about instantiating different cryptographic primitives based on lattices is quite well. However, for improving the efficiency of existent lattice-based construction, e.g. reducing the size of public parameters and ciphertexts, or simplifying the decryption algorithm, our understanding is limited. Besides the theoretical interests in shrinking the size of ciphertext, as the main motivation of studying functional encryption comes from its potential deployment in complex networks and cloud computing, thus the size of transmitted data is a bottleneck of current lattice-based constructions. Combining all these, this brings us to the following open question:

Can we optimize the size of public parameters and ciphertexts of other functional encryption scheme beyond identity based encryption?

1.1 Our Contributions

We positively answer the above question by proposing the first lattice-based compact inner product encryption (IPE). Roughly speaking, in an IPE scheme, the secret key sk is associated with a predicate vector $\boldsymbol{v} \in \mathbb{Z}_q^t$ and the ciphertext is associated with an attribute vector $\boldsymbol{w} \in \mathbb{Z}_q^t$. The decryption works if and only if the inner product $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = 0$. Despite this apparently restrictive structure, inner product predicates can support conjunction, subset and range queries on encrypted data [12], as well as disjunctions, polynomial evaluation, and CNF and DNF formulas [20]. Our construction can be summarized in the following informal theorem:

Theorem 1.1 (Main) *Under the standard Learning with Errors assumption, there is an IPE scheme satisfying weak attribute-hiding property for predicate/attribute vector of length $t = \log n$, where (1) the modulus q is a prime of size polynomial in the security parameter n, (2) ciphertexts consist of a vector in $\mathbb{Z}_q^{2m+1}$, where m is the lattice column dimension, and (3) the public parameters consists two matrices in $\mathbb{Z}_q^{n \times m}$ and a vector in $\mathbb{Z}_q^n$.*

Remark 1.2 Our technique only allows us to prove a weak form of anonymity ("attribute hiding"). Specifically, given a ciphertext ct and a number of keys that do not decrypt ct, the user cannot determine the attribute associated with ct. In the strong form of attribute hiding, the user cannot determine the attribute associated with ct even when given keys that do decrypt ct. The weakened form of attribute hiding we do achieve is nonetheless more that is required for ABE

and should be sufficient for many applications of PE. See Section **2** for more detail.

We can also extend our compact IPE construction to support $t = \mathsf{poly}(n)$-length attribute vectors. Let $t' = t/\log n$, our IPE construction supporting $\mathsf{poly}(n)$-length vectors can be stated in the following corollary:

Corollary 1.3 *Under the standard Learning with Errors assumption, there is an IPE scheme with weak attribute-hiding property supporting predicate/attribute vector of length* $t = \mathsf{poly}(n)$*, where (1) the modulus* q *is a prime of size polynomial in the security parameter* n*, (2) ciphertexts consist of a vector in* $\mathbb{Z}_q^{(t'+1)m+1}$*, where* m *is the lattice column dimension and (3) the public parameters consists* $(t'+1)$ *matrices in* $\mathbb{Z}_q^{n\times m}$ *and a vector in* $\mathbb{Z}_q^n$*.*

In addition to reducing the size of public parameters and ciphertexts, our decryption algorithm is computed in an Single-Instruction-Multiple-Data (SIMD) manner. In prior works [3, 26], the decryption computes the inner product between the predicate vector and ciphertext by (1) decomposing the predicate vector, (2) multiplying-then-adding the corresponding vector bit and ciphertext, entry-by-entry. Our efficient decryption algorithm achieves the inner product by just one vector-matrix multiplication.

1.2 Our Techniques

Our high-level approach to compact inner product encryption from LWE begins by revisiting the first lattice-based IPE construction [3] and the novel fully homomorphic encryption proposed recently by Gentry, Sahai and Waters [17].

The Agrawal-Freeman-Vaikuntanathan IPE. We first briefly review the construction of IPE in [3]. Their construction relies on the algebraic structure of ABB-IBE [2] to solve "lattice matching" problem. Lattice matching means the lattice structure computed in decryption algorithm matches the structure used in key generation, and since the secret key is a *short* trapdoor of the desired lattice, thus the decryption succeeds. To encode a predicate vector $\boldsymbol{v} \in \mathbb{Z}_q^t$ according to [3], the key generation first computes the r-ary decomposition of each entry of $\boldsymbol{v}$ as $v_i = \sum_{j=0}^{k} v_{ij} r^j$, and constructs the $\boldsymbol{v}$-specific lattice as

$$[\mathbf{A}|\mathbf{A}_{\boldsymbol{v}}] = [\mathbf{A}|\sum_{i=1}^{t}\sum_{j=0}^{k} v_{ij}\mathbf{A}_{ij}]$$

by "mixing" a *long* public matrices $(\mathbf{A}, \{\mathbf{A}_{ij}\}) \in \mathbb{Z}_q^{n\times m}$. The secret key $\mathsf{sk}_{\boldsymbol{v}}$ is a short trapdoor of lattice $\Lambda_q^{\perp}([\mathbf{A}|\sum_{i=1}^{t}\sum_{j=0}^{k} v_{ij}\mathbf{A}_{ij}])$. To encode an attribute vector $\boldsymbol{w} \in \mathbb{Z}_q^t$, for $i \in [t], j \in [k]$, construct the $\boldsymbol{w}$-specific vector as

$$\boldsymbol{c}_{ij} = \boldsymbol{s}^{\mathsf{T}}(\mathbf{A}_{ij} + r^j w_i \mathbf{B}) + \mathsf{noise}$$

for a randomly chosen vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ and a public matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$. To reduce the noise growth in the inner produce computation, decryption only needs to multiply-then-add the r-ary representation of v_{ij} to its corresponding $\boldsymbol{c}_{ij}$, as

$$\sum_{i=1}^{t}\sum_{j=0}^{k} v_{ij}\boldsymbol{r}_{ij} = \boldsymbol{s}^{\mathsf{T}}(\sum_{i=1}^{t}\sum_{j=0}^{k} v_{ij}\mathbf{A}_{ij} + \langle \boldsymbol{v}, \boldsymbol{w}\rangle \mathbf{B}) + \mathsf{noise}$$

when $\langle \boldsymbol{v}, \boldsymbol{w}\rangle = 0$, the $(\langle \boldsymbol{v}, \boldsymbol{w}\rangle \mathbf{B})$ part vanishes, thus the lattice computed after inner produce matches the $\mathbf{A}_{\boldsymbol{v}}$ part in the key generation. Then the secret key $\mathsf{sk}_{\boldsymbol{v}}$ can be used to decrypt the ciphertext. Thererfore, the number of matrices in public parameters or vectors in ciphertext is quasilinear in the dimension of vectors.

Using GSW-FHE to compute inner product. Recent progress in fully homomorphic encryption [17] makes us re-think the process of computing inner product. We wonder whether we can use GSW-FHE [17] along with its simplification [5] to simplify the computing procedure. Recall ciphertext of message $x \in \mathbb{Z}_q$ in GSW-FHE can be view in the form $\mathsf{ct}_x = \mathbf{AR} + x\mathbf{G}$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a LWE matrix, $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$ is a random small matrix and $\mathbf{G}$ is the "gadget matrix" as first (explicitly) introduced in the work [21]. The salient point is that there is an efficiently computable function $\mathbf{G}^{-1}$, so that (1) $\mathsf{ct}_x \cdot \mathbf{G}^{-1}(y\mathbf{G}) = \mathsf{ct}_{xy}$, and (2) each entry in matrix $\mathbf{G}^{-1}(y\mathbf{G})$ is just 0 or 1, and thus has small norm. These two nice properties can shrink the size of public parameters (ciphertext) from quasilinear to linear. In particular, to encoding a predicate vector $\boldsymbol{v} \in \mathbb{Z}_q^t$, we construct the $\boldsymbol{v}$-specific lattice as

$$[\mathbf{A}|\mathbf{A}_{\boldsymbol{v}}] = [\mathbf{A}|\sum_{i=1}^{t} \mathbf{A}_i \mathbf{G}^{-1}(v_i\mathbf{G})]$$

where the number of public matrices is $t+1$. To encode an attribute vector $\boldsymbol{w} \in \mathbb{Z}_q^t$, for $i \in [t]$, construct the $\boldsymbol{w}$-specific vector as

$$\boldsymbol{c}_i = \boldsymbol{s}^{\mathsf{T}}(\mathbf{A}_i + w_i\mathbf{G}) + \mathsf{noise}$$

Then, we can compute the inner product as

$$\sum_{i=1}^{t} \boldsymbol{c}_i \cdot \mathbf{G}^{-1}(v_i\mathbf{G}) = \boldsymbol{s}^{\mathsf{T}}(\sum_{i=1}^{t} \mathbf{A}_i\mathbf{G}^{-1}(v_i\mathbf{G}) + \langle \boldsymbol{v}, \boldsymbol{w}\rangle \mathbf{G}) + \mathsf{noise}$$

Since $\mathbf{G}^{-1}(v_i\mathbf{G})$ is small norm, the decryption succeeds when $\langle \boldsymbol{v}, \boldsymbol{w}\rangle = 0$.

Achieving public parameters of two matrices. Our final step is to bring the size of public parameters (or ciphertext) to constant for $(t = \log \lambda)$-length vectors. Inspired by recent work [7] in optimizing size of public parameters in the IBE setting, we use their vector encoding method to further optimize our IPE construction. The vector encoding for encoding $\boldsymbol{v} \in \mathbb{Z}_q^t$ is

$$\mathbf{E}_{\boldsymbol{v}} = \left[v_1\mathbf{I}_n| \cdots |v_t\mathbf{I}_n\right] \cdot \mathbf{G}_{tn,\ell,m}$$

where $\mathbf{G}_{tn,\ell,m} \in \mathbb{Z}_q^{tn \times m}$ is the generalized gadget matrix introduced in [21, 7]. The dimension of this generalized gadget matrix $tn \times tn \log_\ell m$. By setting $t = \log q$ and $\ell = n$, we can obtain the similar column dimension as origin gadget matrix, i.e. $O(n \log q)$. Then the $\boldsymbol{v}$-specific lattice becomes

$$\mathbf{A}_{\boldsymbol{v}} = \mathbf{A}_1 \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(\begin{bmatrix} v_1\mathbf{I}_n \\ \vdots \\ v_d\mathbf{I}_n \end{bmatrix} \cdot \mathbf{G}_{n,2,m} \right)$$

and the $\boldsymbol{w}$-specific ciphertext becomes

$$\boldsymbol{c} = \boldsymbol{s}^\mathsf{T}(\mathbf{A}_1 + \mathbf{E}_{\boldsymbol{w}}) + \mathsf{noise}$$

The inner product can be computed in an SIMD way, as

$$\boldsymbol{c}\cdot\mathbf{G}_{dn,\ell,m}^{-1}\left(\begin{bmatrix} v_1\mathbf{I}_n \\ \vdots \\ v_d\mathbf{I}_n \end{bmatrix} \cdot \mathbf{G}_{n,2,m} \right) \approx \boldsymbol{s}^\mathsf{T}(\mathbf{A}_1 \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(\begin{bmatrix} v_1\mathbf{I}_n \\ \vdots \\ v_d\mathbf{I}_n \end{bmatrix} \mathbf{G}_{n,2,m} \right) + \langle \boldsymbol{v}, \boldsymbol{w} \rangle \mathbf{G}_{n,2,m})$$

As such, our final IPE system contains only two matrices $(\mathbf{A}, \mathbf{A}_1)$ (and a vector $\boldsymbol{u}$), and the ciphertext consists of two vectors. By carefully twisting the vector encoding and proof techniques shown in [3], we show our IPE construction satisfies weakly attribute-hiding. Our IPE system can also be extended in a "parallel repetition" manner to support $(t = \lambda)$-length vectors, as Corollary 1.3 states.

1.3 Related Work

In this section, we provide a comparison with the first IPE construction [3] and its follow-up improvement [26]. In [26], Xagawa used the "Full-Rank Difference encoding", proposed in [2] to map the vector $\mathbb{Z}_q^t$ to a matrix in $\mathbb{Z}_q^{n \times n}$. The size of public parameters (or ciphertext) in his scheme depends linearly on the length of predicate/attribute vectors, and the "Full-Rank Difference encoding" incurs more computation overhead than embedding GSW-FHE structure in IPE construction as described above. The detailed comparison is provided in Table 1.3 for length parameter $t = \log \lambda$.

Table 1. Comparison of Lattice-based IPE Scheme

Schemes	# of $\mathbb{Z}_q^{n \times m}$ mat. in $\lvert\mathsf{pp}\rvert$	# of $\mathbb{Z}_q^m$ vec. in $\lvert\mathsf{ct}\rvert$	LWE param $1/\alpha$
[3]	$O(\lambda \log \lambda)$	$O(\lambda \log \lambda)$	$O(\lambda^{3.5})$
[26]	$O(\lambda)$	$O(\lambda)$	$O(\lambda^4)$
Ours	2	2	$O(\lambda^4 \log \lambda)$

2 Preliminaries

Notation. Let λ be the security parameter, and let PPT denote probabilistic polynomial time. We use bold uppercase letters to denote matrices $\mathbf{M}$, and bold lowercase letters to denote vectors $\boldsymbol{v}$. We write $\widetilde{\mathbf{M}}$ to denote the Gram-Schmidt orthogonalization of $\mathbf{M}$. We write $[n]$ to denote the set $\{1, ..., n\}$, and $|\boldsymbol{t}|$ to denote the number of bits in the string $\boldsymbol{t}$. We denote the i-th bit $\boldsymbol{s}$ by $\boldsymbol{s}[i]$. We say a function $\mathsf{negl}(\cdot) : \mathbb{N} \to (0, 1)$ is negligible, if for every constant $c \in \mathbb{N}$, $\mathsf{negl}(n) < n^{-c}$ for sufficiently large n.

2.1 Inner Product Encryption

We recall the syntax and security definition of *inner product encryption* (IPE) [20, 3]. IPE can be regarded as a generalization of predicate encryption. An IPE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ can be described as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm outputs public parameters pp and master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{v})$: On input the master secret key msk and a predicate vector $\boldsymbol{v}$, the key generation algorithm outputs a secret key $\mathsf{sk}_{\boldsymbol{v}}$ for vector $\boldsymbol{v}$.
- $\mathsf{Enc}(\mathsf{pp}, \boldsymbol{w}, \mu)$: On input the public parameter pp and an attribute/message pair $(\boldsymbol{w}, \mu)$, it outputs a ciphertext $\mathsf{ct}_{\boldsymbol{w}}$.
- $\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{v}}, \mathsf{ct}_{\boldsymbol{w}})$: On input the secret key $\mathsf{sk}_{\boldsymbol{v}}$ and a ciphertext $\mathsf{ct}_{\boldsymbol{w}}$, it outputs the corresponding plaintext μ if $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = 0$; otherwise, it outputs $\perp$.

Definition 2.1 (Correctness) *We say the IPE scheme described above is correct, if for any* $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$*, any message* μ*, any predicate vector* $\boldsymbol{v} \in \mathbb{Z}_q^d$*, and attribute vector* $\boldsymbol{w} \in \mathbb{Z}_q^d$ *such that* $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = 0$*, we have* $\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{v}}, \mathsf{ct}_{\boldsymbol{w}}) = \mu$*, where* $\mathsf{sk}_{\boldsymbol{w}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{v})$ *and* $\mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{Enc}(\mathsf{pp}, \boldsymbol{w}, \mu)$*.*

Security. For the weakly attribute-hiding property of IPE, we use the following experiment to describe it. Formally, for any PPT adversary $\mathcal{A}$, we consider the experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{IPE}}(1^\lambda)$:

- **Setup**: Adversary $\mathcal{A}$ sends two challenge attribute vectors $\boldsymbol{w}_0, \boldsymbol{w}_1 \in \mathbb{Z}_q^d$ to challenger. A challenger runs the $\mathsf{Setup}(1^\lambda)$ algorithm, and sends back the master public key pp.
- **Query Phase I**: Proceeding adaptively, the adversary $\mathcal{A}$ queries a sequence of predicate vectors $(\boldsymbol{v}_1, ..., \boldsymbol{v}_m)$ subject to the restriction that $\langle \boldsymbol{v}_i, \boldsymbol{w}_0 \rangle \neq 0$ and $\langle \boldsymbol{v}_i, \boldsymbol{w}_1 \rangle \neq 0$. On the i-th query, the challenger runs $\mathsf{sk}_{\boldsymbol{v}_i} \to \mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{v}_i)$, and sends the result $\mathsf{sk}_{\boldsymbol{v}_i}$ to $\mathcal{A}$.
- **Challenge**: Once adversary $\mathcal{A}$ decides that Query Phase I is over, he outputs two length-equal messages (μ_0^*, μ_1^*) and sends them to challenger. In response, the challenger selects a random bit $b^* \in \{0, 1\}$, and sends the ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \boldsymbol{w}_{b^*}, \mu_{b^*})$ to adversary $\mathcal{A}$.
- **Query Phase II**: Adversary $\mathcal{A}$ continues to issue secret key queries $(\boldsymbol{v}_{m+1}, ..., \boldsymbol{v}_n)$ adaptively, subject to the restriction that $\langle \boldsymbol{v}_i, \boldsymbol{w}_0 \rangle \neq 0$ and $\langle \boldsymbol{v}_i, \boldsymbol{w}_1 \rangle \neq 0$. The challenger responds by sending back keys $\mathsf{sk}_{\boldsymbol{v}_i}$ as in Query Phase I.

– **Guess**: Adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

We note that query phase I and II can happen polynomial times in terms of security parameter. The advantage of adversary $\mathcal{A}$ in attacking an IPE scheme Π is defined as:

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) = \left| \Pr[b^* = b'] - \frac{1}{2} \right|,$$

where the probability is over the randomness of the challenger and adversary.

Definition 2.2 (Weakly attribute-hiding) *We say an IPE scheme Π is weakly attribute-hiding against chosen-plaintext attacks in selective attribute setting, if for all* PPT *adversaries $\mathcal{A}$ engaging in experiment $\mathbf{Expt}^{\mathsf{IPE}}_{\mathcal{A}}(1^\lambda)$, we have*

$$\mathbf{Adv}_{\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda).$$

2.2 LWE and Sampling Algorithms over Lattices

Learning With Errors. The LWE problem was introduced by Regev [24], the works of [24] show that the LWE assumption is as hard as (quantum) solving GapSVP and SIVP under various parameter regimes.

Definition 2.3 (LWE) *For an integer $q = q(n) \geq 2$, and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the* Learning With Errors problem $\mathsf{LWE}_{n,m,q,\chi}$ *is to distinguish between the following pairs of distributions (e.g. as given by a sampling oracle $\mathcal{O} \in \{\mathcal{O}_{\boldsymbol{s}}, \mathcal{O}_{\$}\}$):*

$$\{\mathbf{A}, \boldsymbol{s}^\mathsf{T}\mathbf{A} + \boldsymbol{x}^\mathsf{T}\} \text{ and } \{\mathbf{A}, \boldsymbol{u}\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^m$, and $\boldsymbol{x} \xleftarrow{\$} \chi^m$.

Two-Sided Trapdoors and Sampling Algorithms. We will use the following algorithms to sample short vectors from specified lattices.

Lemma 2.4 ([16, 6]) *Let q, n, m be positive integers with $q \geq 2$ and sufficiently large $m = \Omega(n \log q)$. There exists a* PPT *algorithm $\mathsf{TrapGen}(q, n, m)$ that with overwhelming probability outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m\times m})$ such that $\mathbf{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and $\mathbf{T}_{\mathbf{A}}$ is a basis for $\Lambda_q^\perp(\mathbf{A})$ satisfying*

$$||\mathbf{T}_{\mathbf{A}}|| \leq O(n \log q) \quad \text{and} \quad ||\widetilde{\mathbf{T}_{\mathbf{A}}}|| \leq O(\sqrt{n \log q})$$

except with $\mathsf{negl}(n)$ probability.

Lemma 2.5 ([16, 13, 2]) *Let $q > 2, m > n$. There are two sampling algorithms as follows:*

– *There is a* PPT *algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T}_{\mathbf{A}}, \boldsymbol{u}, s)$, taking as input: (1) a rank-n matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and any matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m_1}$, (2) a "short" basis $\mathbf{T}_{\mathbf{A}}$ for lattice $\Lambda_q^\perp(\mathbf{A})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, (3) a Gaussian parameter $s > ||\widetilde{\mathbf{T}_{\mathbf{A}}}|| \cdot \omega(\sqrt{\log(m + m_1)})$. Then outputs a vector $\boldsymbol{r} \in \mathbb{Z}^{m+m_1}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$ where $\mathbf{F} := [\mathbf{A}|\mathbf{B}]$.*

– *There is a* PPT *algorithm* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \boldsymbol{u}, s)$*, taking as input: (1) a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$*, and a rank-*$n$ *matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$*, a matrix* $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$*, where* $s_{\mathbf{R}} := ||\mathbf{R}|| = \sup_{\boldsymbol{x}:||\boldsymbol{x}||=1} ||\mathbf{R}\boldsymbol{x}||$*, (2) a "short" basis* $\mathbf{T_B}$ *for lattice* $\Lambda_q^{\perp}(\mathbf{B})$*, a vector* $\boldsymbol{u} \in \mathbb{Z}_q^n$*, (3) a Gaussian parameter* $s > ||\widetilde{\mathbf{T_B}}|| \cdot s_{\mathbf{R}} \cdot \omega(\sqrt{\log m})$*. Then outputs a vector* $\boldsymbol{r} \in \mathbb{Z}^{2m}$ *distributed statistically close to* $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$ *where* $\mathbf{F} := (\mathbf{A}|\mathbf{AR} + \mathbf{B})$.

Gadget Matrix. We now recall the gadget matrix [21, 5], and the extended gadget matrix technique appeared in [7], that are important to our construction.

Definition 2.6 *Let* $m = n \cdot \lceil \log q \rceil$*, and define the gadget matrix*

$$\mathbf{G}_{n,2,m} = \boldsymbol{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times m}$$

where vector $\boldsymbol{g} = (1, 2, 4, ..., 2^{\lfloor \log q \rfloor}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$*, and* $\otimes$ *denotes tenser product. We will also refer to this gadget matrix as "powers-of-two" matrix. We define the inverse function* $\mathbf{G}_{n,2,m}^{-1} : \mathbb{Z}_q^{n \times m} \to \{0,1\}^{m \times m}$ *which expands each entry* $a \in \mathbb{Z}_q$ *of the input matrix into a column of size* $\lceil \log q \rceil$ *consisting of the bits of binary representations. We have the property that for any matrix* $\mathbf{A} \in Z_q^{n \times m}$*, it holds that* $\mathbf{G}_{n,2,m} \cdot \mathbf{G}_{n,2,m}^{-1}(\mathbf{A}) = \mathbf{A}$.

As mentioned by [21] and explicitly described in [7], the results for $\mathbf{G}_{n,2,m}$ and its trapdoor can be extended to other integer powers or mixed-integer products. In this direction, we give a generalized notation for gadget matrices as follows:

3 Our Construction

In this section, we describe our compact IPE construction. Before diving into the details, we first revisit a novel encoding method implicitly employed in adaptively secure IBE setting in [7]. Consider the vector space $\mathbb{Z}_q^d$. For vector $\boldsymbol{v} = (v_1, ..., v_d) \in \mathbb{Z}_q^d$, we define the following encoding algorithm which maps a d-dimensional vector to an $n \times m$ matrix.

$$\mathsf{encode}(\boldsymbol{v}) = \mathbf{E}_{\boldsymbol{v}} = \left[v_1\mathbf{I}_n | \cdots | v_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m} \tag{1}$$

Similarly, we also define the encoding for an integer $a \in \mathbb{Z}_q$ as: $\mathsf{encode}(a) = \mathbf{E}_a = a\mathbf{G}_{n,2,m}$. The above encoding supports the vector space operations naturally, and our compact IPE construction relies on this property.

3.1 IPE Construction Supporting log(λ)-length Attributes

We describe our IPE scheme that each secret key is associated with a predicate vector $\boldsymbol{v} \in \mathbb{Z}_q^d$ (for some fixed $d = \log \lambda$), and each ciphertext will be associated with an attribute vector $\boldsymbol{w} \in \mathbb{Z}_q^d$. Decryption succeeds if and only if $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = 0 \bmod q$. We further extend our IPE construction supporting $d = \mathsf{poly}(\lambda)$-length vectors in Section 3.3. The description of $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows:

– $\mathsf{Setup}(1^\lambda, 1^d)$: On input the security parameter λ and length parameter d, the setup algorithm first sets the parameters (q, n, m, s) as below. We assume the parameters (q, n, m, s) are implicitly included in both pp and msk. Then it generates a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ along with its trapdoor $\mathbf{T}_\mathbf{A} \in \mathbb{Z}_q^{m \times m}$, using $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(q, n, m)$. Next sample a random matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. Output the public parameter pp and master secret key msk as

$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \boldsymbol{u}), \qquad \mathsf{msk} = (\mathsf{pp}, \mathbf{T}_\mathbf{A})$$

– $\mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{v})$: On input the master secret key msk and predictor vector $\boldsymbol{v} = (v_1, ..., v_d) \in \mathbb{Z}_q^d$, the key generation algorithm first sets matrix $\mathbf{B}_\boldsymbol{v}$ as

$$\mathbf{B}_\boldsymbol{v} = \mathbf{B} \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(\begin{bmatrix} v_1\mathbf{I}_n \\ \vdots \\ v_d\mathbf{I}_n \end{bmatrix} \cdot \mathbf{G}_{n,2,m} \right)$$

Then sample a low-norm vector $\boldsymbol{r}_\boldsymbol{v} \in \mathbb{Z}^{2m}$ using algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}_\boldsymbol{v}, \boldsymbol{u}, s)$, such that $[\mathbf{A}|\mathbf{B}_\boldsymbol{v}] \cdot \boldsymbol{r}_\boldsymbol{v} = \boldsymbol{u} \bmod q$. Output secret key $\mathsf{sk}_\boldsymbol{v} = \boldsymbol{r}_\boldsymbol{v}$.

– $\mathsf{Enc}(\mathsf{pp}, \boldsymbol{w}, \mu)$: On input the public parameter pp, an attribute vector $\boldsymbol{w} = (w_1, ..., w_d) \in \mathbb{Z}_q^d$ and a message $\mu \in \{0, 1\}$, the encryption algorithm first chooses a random vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ and a random matrix $\mathbf{R} \in \{-1, 1\}^{m \times m}$. Then encode the attribute vector $\boldsymbol{w}$ as in Equation (1)

$$\mathbf{E}_\boldsymbol{w} = \left[w_1\mathbf{I}_n | \cdots | w_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m}$$

Let the ciphertext $\mathsf{ct}_\boldsymbol{w} = (\boldsymbol{c}_0, \boldsymbol{c}_1, c_2) \in \mathbb{Z}_q^{2m+1}$ be

$$\boldsymbol{c}_0 = \boldsymbol{s}^\mathsf{T}\mathbf{A} + \boldsymbol{e}_0^\mathsf{T}, \quad \boldsymbol{c}_1 = \boldsymbol{s}^\mathsf{T}(\mathbf{B} + \mathbf{E}_\boldsymbol{w}) + \boldsymbol{e}_0^T\mathbf{R}, \quad c_2 = \boldsymbol{s}^\mathsf{T}\boldsymbol{u} + e_1 + \lceil q/2 \rceil \mu$$

where errors $\boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}, e_1 \leftarrow \mathcal{D}_{\mathbb{Z},s}$.

– $\mathsf{Dec}(\mathsf{sk}_\boldsymbol{v}, \mathsf{ct}_\boldsymbol{w})$: On input the secret key $\mathsf{sk}_\boldsymbol{v} = \boldsymbol{r}_\boldsymbol{v}$ and ciphertext $\mathsf{ct}_\boldsymbol{w} = (\boldsymbol{c}_0, \boldsymbol{c}_1, c_2)$, if $\langle \boldsymbol{v}, \boldsymbol{w} \rangle \neq 0 \bmod q$, then output $\perp$. Otherwise, first compute

$$\boldsymbol{c}_1' = \boldsymbol{c}_1 \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(\begin{bmatrix} v_1\mathbf{I}_n \\ \vdots \\ v_d\mathbf{I}_n \end{bmatrix} \cdot \mathbf{G}_{n,2,m} \right)$$

then output $\mathsf{Round}(c_2 - \langle (\boldsymbol{c}_0, \boldsymbol{c}_1'), \boldsymbol{r}_\boldsymbol{v} \rangle)$.

Lemma 3.1 *The IPE scheme Π described above is correct (c.f. Definition 2.1).*

Proof. When the predicate vector $\boldsymbol{v}$ and attribute vector $\boldsymbol{w}$ satisfies $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = 0 \bmod q$, it holds that $\boldsymbol{c}_1' = \boldsymbol{s}^\mathsf{T}\mathbf{B}_\boldsymbol{v} + \boldsymbol{e}_0'$. Therefore, during decryption, we have

$$\mu' = \mathsf{Round}\left(\lceil q/2 \rceil \mu + \underbrace{e_1 - \langle (\boldsymbol{e}_0, \boldsymbol{e}_0'), \boldsymbol{r}_\boldsymbol{v} \rangle}_{\text{small}} \right) = \mu \in \{0, 1\}$$

The third equation follows if $(e_1 - \langle (\boldsymbol{e}_0, \boldsymbol{e}_0'), \boldsymbol{r}_\boldsymbol{v} \rangle)$ is indeed small, which holds w.h.p. by setting the parameters appropriately below. □

Parameter Selection. To support $d = \log(\lambda)$-length predicate/attribute vectors, we set the system parameters according to Table 2, where $\epsilon > 0$ is an arbitrarily small constant.

Parameters	Description	Setting
λ	security parameter	
n	lattice row dimension	λ
m	lattice column dimension	$n^{1+\epsilon}$
q	modulus	$n^{3+\epsilon}m$
s	sampling and error width	$n^{1+\epsilon}$
ℓ	integer-base parameter	n

Table 2. $\log(\lambda)$-length IPE Parameters Setting

These values are chosen in order to satisfy the following constraints:

- To ensure correctness, we require $|e_1 - \langle(\boldsymbol{e}_0, \boldsymbol{e}_0'), \boldsymbol{r_v}\rangle| < q/4$; Let $\boldsymbol{r_v} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$, here we can bound the dominating term:

$$|\boldsymbol{e}_0^{'\mathsf{T}} \boldsymbol{r}_2| \leq ||\boldsymbol{e}_0^{'\mathsf{T}}|| \cdot ||\boldsymbol{r}_2|| \approx s\sqrt{m}d\ell \log_\ell q \cdot s\sqrt{m} = s^2 m n^{1+\epsilon} < q/4$$

- For $\mathsf{SampleLeft}$, we know $||\widetilde{\mathbf{T}_\mathbf{A}}|| = O(\sqrt{n\log(q)})$, thus this requires that the sampling width s satisfies $s > \sqrt{n\log(q)} \cdot \omega(\sqrt{\log(m)})$. For $\mathsf{SampleRight}$, we need $s > ||\widetilde{\mathbf{T}_{\mathbf{G}_{n,2,m}}}|| \cdot ||\mathbf{R}||\omega(\sqrt{\log m}) = n^{1+\epsilon}\omega(\sqrt{\log m})$. To apply Regev's reduction, we need $s > \sqrt{n}\omega(\log(n))$ (s here is an absolute value, not a ratio). Therefore, we need $s > n^{1+\epsilon}$
- To apply the Leftover Hash Lemma, we need $m \geq (n+1)\log(q) + \omega(\log(n))$.

3.2 Security Proof

In this part, we show the weakly attribute-hiding property of our IPE construction. We adapt the simulation technique in [3] by plugin the encoding of vectors. Intuitively, to prove the theorem we define a sequences of hybrids against adversary $\mathcal{A}$ in the weak attribute-hiding experiment. The adversary $\mathcal{A}$ outputs two attribute vectors $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$ at the beginning of each game, and at some point outputs two messages μ_0, μ_1. The first and last games correspond to real security game with challenge ciphertexts $\mathsf{Enc}(\mathsf{pp}, \boldsymbol{w}_0, \mu_0)$ and $\mathsf{Enc}(\mathsf{pp}, \boldsymbol{w}_1, \mu_1)$ respectively. In the intermediate games we use the "alternative" simulation algorithms $(\mathsf{Sim.Setup}, \mathsf{Sim.KeyGen}, \mathsf{Sim.Enc})$. During the course of the game the adversary can only request keys for predicate vector $\boldsymbol{v}_i$ such that $\langle \boldsymbol{v}_i, \boldsymbol{w}_0\rangle \neq 0$ and $\langle \boldsymbol{v}_i, \boldsymbol{w}_1\rangle \neq 0$. We first define the simulation algorithms $(\mathsf{Sim.Setup}, \mathsf{Sim.KeyGen}, \mathsf{Sim.Enc})$ in the following:

- $\mathsf{Sim.Setup}(1^\lambda, 1^d, \boldsymbol{w}^*)$: On input the security parameter λ, the length parameter d, and an attribute vector $\boldsymbol{w}^* \in \mathbb{Z}_q^d$, the simulation setup algorithm first

chooses a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and a random vector $\boldsymbol{u} \leftarrow \mathbb{Z}_q^n$. Then set matrix

$$\mathbf{B} = \mathbf{A}\mathbf{R}^* - \mathbf{E}_{\boldsymbol{w}^*}, \quad \mathbf{E}_{\boldsymbol{w}^*} = \left[w_1^*\mathbf{I}_n | \cdots | w_d^*\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m}$$

where matrix $\mathbf{R}^*$ is chosen randomly from $\{-1,1\}^{m \times m}$. Output $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \boldsymbol{u})$ and $\mathsf{msk} = \mathbf{R}^*$.

– $\mathsf{Sim.KeyGen}(\mathsf{msk}, \boldsymbol{v})$: On input the master secret key msk and a vector $\boldsymbol{v} \in \mathbb{Z}_q^d$, the simulation key generation algorithm sets matrix $\mathbf{R}_{\boldsymbol{v}}$ and $\mathbf{B}_{\boldsymbol{v}}$ as

$$\mathbf{R}_{\boldsymbol{v}} = \left(\begin{bmatrix} v_1\mathbf{I}_n \\ \vdots \\ v_d\mathbf{I}_n \end{bmatrix} \cdot \mathbf{G}_{n,2,m} \right), \quad \mathbf{B}_{\boldsymbol{v}} = \mathbf{B} \cdot \mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{R}_{\boldsymbol{v}})$$

Then sample a low-norm vector $\boldsymbol{r}_{\boldsymbol{v}} \in \mathbb{Z}^{2m}$ using algorithm

$$\boldsymbol{r}_{\boldsymbol{v}} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \langle \boldsymbol{v}, \boldsymbol{w}^* \rangle \mathbf{G}_{n,2,m}, \mathbf{R}^*\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{R}_{\boldsymbol{v}}), \mathbf{T}_{\mathbf{G}_{n,2,m}}\boldsymbol{u}, s)$$

such that $[\mathbf{A}|\mathbf{B}_{\boldsymbol{v}}] \cdot \boldsymbol{r}_{\boldsymbol{v}} = \boldsymbol{u} \bmod q$. Output secret key $\mathsf{sk}_{\boldsymbol{v}} = \boldsymbol{r}_{\boldsymbol{v}}$.

– $\mathsf{Sim.Enc}(\mathsf{pp}, \boldsymbol{w}^*, \mu)$: The simulation encryption algorithm is the same as the counterpart in the scheme, except the matrix $\mathbf{R}^*$ is used in generating the ciphertext instead of sampling a random matrix $\mathbf{R} \in \{-1,1\}^{m \times m}$.

Due to the space limit, we include proof of the following theorem in full version.

Theorem 3.2 *Assuming the hardness of (n, q, χ)-LWE assumption, the IPE scheme described above is weakly attribute-hiding (c.f. Definition 2.2).*

3.3 IPE Construction Supporting poly(λ)-length Vectors

We also extend our IPE construction to support $t = \mathsf{poly}(\lambda)$-length vectors, which means the predicate and attribute vector are chosen in vector space $\mathbb{Z}_q^t$. Intuitively speaking, our construction described below can be regarded as a $t' = \lceil t/d \rceil$ "parallel repetition" version of IPE construction for $d = \log(\lambda)$-length vectors. In particular, we encode every $\log(\lambda)$ part of the attribute vector $\boldsymbol{v}$, and then concatenate these encoding together as the encoding of $\boldsymbol{v}$. Due to space limit, we include the detailed scheme and proof in the full version.

References

1. Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *Crypto 2017*.
2. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT 2010*.
3. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT 2011*,
4. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

5. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO 2014, Part I*.
6. Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2010.
7. Daniel Apon, Xiong Fan, and Feng-Hao Liu. Vector encoding over lattices and its applications. Cryptology ePrint Archive, Report 2017/455, 2017.
8. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*.
9. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*.
10. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT 2014*.
11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*.
12. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*.
13. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*.
14. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*.
15. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, 2009.
16. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, May 2008.
17. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013, Part I*.
18. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *45th ACM STOC*, 2013.
19. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 06*.
20. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*.
21. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*.
22. Michael Mitzenmacher, editor. *41st ACM STOC*. ACM Press, May / June 2009.
23. Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/2010/556`.
24. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, May 2005.
25. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*.
26. Keita Xagawa. Improved (hierarchical) inner-product encryption from lattices. In *PKC 2013*.