On the Untapped Potential of Encoding Predicates by Arithmetic Circuits and Their Applications

Shuichi Katsumata *

Abstract

Predicates are used in cryptography as a fundamental tool to control the disclosure of secrets. However, how to embed a particular predicate into a cryptographic primitive is usually not given much attention. In this work, we formalize the idea of encoding predicates as arithmetic circuits and observe that choosing the right encoding of a predicate may lead to an improvement in many aspects such as the efficiency of a scheme or the required hardness assumption. In particular, we develop two predicate encoding schemes with different properties and construct cryptographic primitives that benefit from these: verifiable random functions (VRFs) and predicate encryption (PE) schemes.

- We propose two VRFs on bilinear maps. Both of our schemes are secure under a non-interactive Q-type assumption where Q is only poly-logarithmic in the security parameter, and they achieve either a poly-logarithmic verification key size or proof size. This is a significant improvement over prior works, where all previous schemes either require a strong hardness assumption or a large verification key and proof size.
- We propose a lattice-based PE scheme for the class of *multi-dimensional equality* ($\mathsf{MultD\text{-}Eq}$) predicates. This class of predicate is expressive enough to capture many of the appealing applications that motivates PE schemes. Our scheme achieves the best in terms of the required approximation factor for LWE (we only require $\mathsf{poly}(\lambda)$) and the decryption time. In particular, all existing PE schemes that support the class of $\mathsf{MultD\text{-}Eq}$ predicates either require a subexponential LWE assumption or an exponential decryption time (in the dimension of the $\mathsf{MultD\text{-}Eq}$ predicates).

1 Introduction

A *predicate* is a function $P : \mathcal{X} \to \{0, 1\}$ that partitions an input domain $\mathcal{X}$ into two distinct sets according to some relation. Due to its natural compatibility with cryptographic primitives, predicates have been used in many scenarios to control the disclosure of secrets. This may either come up explicitly during construction (e.g., attribute-based encryptions [SW05, GPSW06], predicate encryptions [BW07, SBC+07, KSW08]) or implicitly during security proofs (e.g., in the form of programmable hashes [HK08, ZCZ16], admissible hashes [BB04a, CHKP10]). However, how to express predicates as (arithmetic) circuits is usually not given much attention in these works. Since the way we embed predicates into a cryptographic primitive has a direct effect on the concrete efficiency of the schemes, it is important to know how efficiently we can embed predicates. In this paper, we propose an efficient encoding for a specific class of predicates and focus on two primitives that benefit from this: verifiable random functions (VRFs) and predicate encryptions (PE) schemes.

*The University of Tokyo, National Institute of Advanced Industrial Science and Technology (AIST).

Verifiable Random Functions. VRFs introduced by Micali, Rabin and Vadhan [MRV99] are a special form of pseudorandom functions (PRFs), which additionally enables a secret key holder to create a non-interactive and publicly verifiable proof that validates the output value. An attractive property for the VRF to have is the notion of *all the desired properties* coined by [HJ16], which captures the following features: an exponential-sized input space, adaptive pseudorandomness, and security under a non-interactive complexity assumption.

There currently exist two approaches for constructing VRFs with all the desired properties. The first approach is to use a specific number theory setting (mainly bilinear groups) to hand-craft VRFs [HW10, BMR10, ACF14, Jag15, HJ16, Yam17], and the second approach is to use a more generic approach and build VRFs from general cryptographic primitives [GHKW17, Bit17, BGJS17]. While the second approach provides us with better insight on VRFs and allows us to base security on hardness assumptions other than bilinear map based ones, the major drawback is the need for large verification key / proof sizes or the need for strong hardness assumptions such as the subexponential Learning with Errors (LWE) assumption to instantiate the underlying primitives. Concretely, all generic approaches require general non-interactive witness indistinguishable proofs (NIWIs) and constrained PRFs for admissible hash friendly functions, which we currently do not know how to simultaneously construct compactly and base security under a weak hardness assumption.

The first approach is more successful overall in light of compactness and the required hardness assumptions, however, they come with their own shortcomings. Notably, [Yam17] presents three constructions where only $\omega(\log \lambda)$ group elements[1] are required for either the verification key or the proof. In particular, in one of their schemes, only sub-linear group elements are required for both verification key and proof. However, all three schemes require an L-DDH[2] assumption where $L = \tilde{\Omega}(\lambda)$. In contrast, [Jag15] presents a scheme secure under a much weaker L-DDH assumption where $L = O(\log \lambda)$ and [HJ16] under the DLIN assumption. However, these approaches require a linear number of group elements in the verification key and proof in the security parameter. Therefore, we currently do not know how to construct VRFs that are both compact and secure under a weak hardness assumption.

Predicate Encryption. A predicate encryption (PE) scheme [BW07, SBC+07, KSW08] is a paradigm for public-key encryption that supports searching on encrypted data. In predicate encryption, ciphertexts are associated with some attribute X, secret keys are associated with some predicate P, and the decryption is successful if and only if $P(X) = 1$. The major difficulty of constructing predicate encryption schemes stems from the security requirement that enforces the privacy of the attribute X and the plaintext even amidst multiple secret key queries.

Some of the motivating applications for predicate encryption schemes that are often stated in the literatures are: inspection of recorded log files for network intrusions, credit card fraud investigation and conditional disclosure of patient records. Notably, all the above applications only require checking whether a subset or range conjunction predicate is satisfied. (For a more thorough discussion, see [BW07, SBC+07, KSW08].) Therefore, in some sense many of the applications that motivates for predicate encryption schemes can be implemented by predicate encryption schemes for the class of predicates that are expressive enough to support subset or range conjunctions.

On the surface, the present situation on lattice-based predicate encryption schemes seem bright. We have concrete constructions based on LWE for the class of predicates that supports

[1] Here, $\omega(f(\lambda))$ denotes any function that grows asymptotically faster than $f(\lambda)$, e.g., $\log^2 \lambda = \omega(\log \lambda)$

[2] The L-DDH problem is where we are given $(h, g, g^{\alpha}, \cdots, g^{\alpha^L}, \Psi)$ and have to decided whether $\Psi = e(g, h)^{1/\alpha}$ or a uniform random element.

equality [ABB10, CHKP10], inner-products [AFV11], multi-dimensional equality (MultD-Eq)[3] [GMW15], and all circuits [GVW15, GKW17, WZ17][4]. Therefore, in theory, we can realize all the above applications in a secure manner, since subset or range conjunctions can be efficiently encoded by any predicate as expressive as the MultD-Eq predicate, i.e., the works of [GMW15, GVW15, GKW17, WZ17] are all sufficient for the above applications. However, all of these schemes may be too inefficient to use in real-life applications. Namely, the scheme of [GMW15] highly resembles the bilinear map based construction of [SBC+07] and inherits the same problem; it takes $\Omega(2^D)$ decryption time where D roughly corresponds to the number of set elements specifying the subset predicate or the number of conjunctions used in the range conjunction predicate. Further, the schemes of [GVW15, GKW17, WZ17] are powerful and elegant, albeit they all require subexponential LWE assumptions. Therefore, aiming at predicate encryption schemes with the above applications in mind, we currently do not have satisfactorily efficient lattice-based schemes. In particular, we do not know how to construct efficient lattice-based PE schemes for the class of MultD-Eq predicates. This is in sharp contrast with the bilinear map setting where we know how to obtain efficient schemes for the above applications [BW07].

1.1 Our Contributions

In this paper, we provide two results: a compact VRF under a weak assumption and an efficient lattice-based PE scheme for the class of MultD-Eq predicates. For the time being, it suffices to think of the MultD-Eq predicate as simply a predicate that supports the subset predicate. Here, although the two results may seem independent, they are in fact related by a common theme that they both implicitly or explicitly embed the subset predicates in their constructions.

Our idea is simple. We first detach predicates from cryptographic constructions, and view predicates simply as a function. Then, we introduce the notion of *predicate encoding schemes*[5], where we encode predicates as simple (arithmetic) circuits that have different properties fit for the underlying cryptographic applications. For example, we might not care that a predicate P outputs 0 or 1. We may only care that P behaves differently on satisfied/non-satisfied inputs, e.g., P outputs a value in S_0 when it is satisfied and S_1 otherwise, where S_0, S_1 are disjoint sets. In particular, we provide two predicate encoding schemes $\mathsf{PES}_{\mathsf{FP}}$ and $\mathsf{PES}_{\mathsf{Lin}}$ with different properties encoding the MultD-Eq predicates. Then, based on these encoded MultD-Eq predicates, we construct our VRFs, and PE schemes for the class of MultD-Eq predicates. The following is a summary of our two results.

VRF. We propose two VRFs with all the desired properties. A detailed comparison is provided in Table 1. Note that we intentionally excluded the recent VRF constructions of [Bit17, BGJS17, GHKW17] from the table, since their schemes cannot be instantiated efficiently due to the lack of efficient (general) NIWIs and constrained PRFs.

[3] The precise definition and discussions of this predicate are given in Sec. 4.2. For the time being, it is enough to view it as a subset predicate.

[4] [GKW17, WZ17] give a generic conversion from ABEs to PEs that uses an obfuscation for a specific program proven secure under the subexponential LWE assumption. Therefore, we have provably secure lattice-based PEs for all circuits using the lattice-based ABE of [GVW13, BGG+14].

[5] We note that the term "predicate encoding" has already been used in a completely different context by [Wee14]. See the section of related work for the differences.

Table 1: Comparison of VRFs with all the desired properties.

Schemes	$\lvert\mathsf{vk}\rvert$ (# of $\mathbb{G}$)	$\lvert\mathsf{sk}\rvert$ (# of $\mathbb{Z}_p$)	$\lvert\pi\rvert$ (# of $\mathbb{G}$)	Assumption	Reduction Cost
[BMR10]	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$-DDH	$O(\epsilon/\lambda)$
[HW10]	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	$O(\lambda Q/\epsilon)$-DDHE	$O(\epsilon^2/\lambda Q)$
[ACF14]	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})^*$
[Jag15]	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	$O(\log(Q/\epsilon))$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})^*$
[HJ16]	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	DLIN	$O(\epsilon^{\nu+1}/\lambda Q^{\nu})^*$
[Yam17]: Sec. 7.1.	$\omega(\lambda \log \lambda)$	$\omega(\log \lambda)$	$\omega(\log \lambda)$	$\omega(\lambda \log \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})^*$
[Yam17]: Sec. 7.3.	$\omega(\log \lambda)$	$\omega(\log \lambda)$	$\omega(\sqrt{\lambda} \log \lambda)$	$\omega(\lambda \log \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})^*$
[Yam17]: App. C.	$\omega(\log \lambda)$	$\omega(\log \lambda)$	$\mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$-DDH	$O(\epsilon^2/\lambda^2 Q)$
Ours: Sec. 5.2.	$\omega(\log^2 \lambda)$	$\omega(\log^2 \lambda)$	$\omega(\lambda \log^2 \lambda)$	$\omega(\log^2 \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})^*$
Ours: Sec. 5.4.	$\omega(\sqrt{\lambda} \log \lambda)$	$\omega(\log^2 \lambda)$	$\omega(\log \lambda)$	$\omega(\log^2 \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})^*$

To measure the verification key size $\lvert\mathsf{vk}\rvert$ and proof size $\lvert\pi\rvert$ (resp. secret key size $\lvert\mathsf{sk}\rvert$), we count the number of group elements in $\mathbb{G}$ (resp. $\mathbb{Z}_p$). Q, ϵ denotes the number of adversarial queries and advantage, respectively. We measure all the reduction cost using the techniques of [BR09]. $\omega(f(\lambda))$ means that it can be taken as any function that grows asymptotically faster than $f(\lambda)$; for simplicity one can instead interpret the above $\omega(f(\lambda))$ terms as $O(\log \lambda \cdot f(\lambda))$. $\mathsf{poly}(\lambda)$ represents a fixed polynomial that does not depend on Q, ϵ.

* ν is a constant satisfying $c = 1 - 2^{-1/\nu}$, where c is the relative distance of the underlying error correcting code $C : \{0,1\}^n \to \{0,1\}^\ell$. We can make ν arbitrary close to 1 by choosing $c < 1/2$ appropriately and setting ℓ large enough. (For further detail, see [Gol08], Appendix E.1)

Our constructions are inspired by the bilinear map based VRFs of [Yam17], where they noticed that an admissible hash function [BB04b, CHKP10] can be represented much more compactly by using a subset predicate[6]. We improve their works by further noticing that subset predicates, when viewed as simply a function, can be encoded in various ways into a circuit. In particular, we propose a more efficient circuit encoding ($\mathsf{PES}_{\mathsf{FP}}$) of the subset predicates that is compatible with the underlying algebraic structure of the VRF. We note that at the technical level the constructions are quite different; [Yam17] uses the inversion-based techniques [DY05, BMR10] whereas we do not. Here, simply using $\mathsf{PES}_{\mathsf{FP}}$ already provides us with an improvement over previous schemes, however, by exploiting a special linear structure in $\mathsf{PES}_{\mathsf{FP}}$, we can further improve the efficiency using an idea native to our scheme. Namely, we can skip some of the verification steps required to check the validity of the proof, hence, lowering the number of group elements in the verification key. Our schemes can be viewed as combining the best of [Jag15] and [Yam17]. In the following, to compare the efficiency, we count the number of group elements of the verification key and proof.

- In our first scheme, the verification key size is $\omega(\log^2 \lambda)$, the proof size is $\omega(\lambda \log^2 \lambda)$, and the scheme is proven secure under the L-DDH assumption with $L = \omega(\log^2 \lambda)$. This is the first scheme that simultaneously achieves a small verification key size and security under an L-DDH assumption where L is poly-logarithm in the security parameter.

- Our second scheme is a modification of our first VRF with some additional ideas; the verification key size is $\omega(\sqrt{\lambda} \log \lambda)$, the proof size is $\omega(\log \lambda)$, and the scheme is proven secure under the L-DDH assumption with $L = \omega(\log^2 \lambda)$. This achieves the smallest verification

[6] In particular, our idea is inspired by the VRFs based on the admissible hash function of [Yam17], Sec. 6. However, the construction is more similar to the VRF based on the variant of Water's hash in their Appendix C.

key and proof size among all the previous schemes while also reducing the underlying L of the L-DDH assumption significantly to poly-logarithm.

PE Schemes for the MultD-Eq Predicates. Based on the predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}}$ for the MultD-Eq predicates, we propose a lattice-based PE scheme for the MultD-Eq predicates. Due to the symmetry of the MultD-Eq predicates, we obtain key-policy and ciphertext-policy predicate encryption schemes for the class of predicates that can be expressed as MultD-Eq, such as subset and range conjunction. A detailed comparison is provided in Table 2. Note that we exclude the generic constructions of [GVW15, GKW17, WZ17] to keep the presentation simple. Although the generic constructions are very powerful and elegant, they all require subexponential LWE even if we restrict the underlying ABE to simple circuit classes and determining the concrete efficiency is not obvious, e.g., in [GKW17] the secret key size depends on the underlying FHE scheme.

Table 2: Comparison of lattice PEs for MultD-Eq predicates (over $\mathbb{Z}_p^{D\times\ell}$).

Schemes	$\lvert\mathsf{mpk}\rvert$ (# of $\mathbb{Z}_q^{n\times m}$)	$\lvert\mathsf{sk}\rvert$ (# of $\mathbb{Z}^{2m}$)	$\lvert\mathsf{ct}\rvert$ (# of $\mathbb{Z}_q^{m}$)	LWE param $1/\alpha$	Dec. Time (# of IP)
[GMW15]	$O(D\ell)$	$O(D\ell)$	$O(D\ell)$	$\tilde{O}(\sqrt{D}\cdot n^{1.5})^{\dagger}$	$O(\ell^D)$
Ours: Sec. 6.2	$O(D\ell p)$	1	$O(D\ell p)$	$\tilde{O}(\max\{\frac{n^2}{\sqrt{D\ell p}}, \sqrt{D\ell p}\cdot n\})$	1

To compare (space) efficiency, we measure the master public key size $\lvert\mathsf{mpk}\rvert$, secret key size $\lvert\mathsf{sk}\rvert$ and ciphertext size $\lvert\mathsf{ct}\rvert$ by the required number of elements in $\mathbb{Z}_q^{n\times m}, \mathbb{Z}^{2m}, \mathbb{Z}_q^m$, respectively. We measure the decryption time as the number of inner products computed between vectors in $\mathbb{Z}_q^{2m}$.

† To be fair, we provided a more rigorous analysis for their parameter selections (as we did with our scheme).

Our scheme achieves the best efficiency in terms of decryption time and the required modulus size q; recall [GMW15] needs to perform $\Omega(2^D)$ number of inner product operations (between secret key vectors and ciphertext vectors) to decrypt a ciphertext, and [GVW15, GKW17, WZ17] require subexponential LWE for security. Furthermore, compared with [GMW15], the number of secret keys (i.e., vectors in $\mathbb{Z}^{2m}$) we require are only one, whereas they require at least $O(D)$. Our construction follows very naturally from the predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}}$ for the MultD-Eq predicates, and builds upon the proof techniques of [AFV11, BGG+14].

Other Applications. We also show how to make the identity-based encryption (IBE) scheme of [Yam17] more efficient by using our predicate encoding scheme for the MultD-Eq predicate. In particular, we are able to lower the approximation factor of the LWE problem from $\tilde{O}(n^{11})$ to $\tilde{O}(n^{5.5})$ (with some additional analysis). Furthermore, we are able to significantly reduce the parallel complexity of the matrix multiplications required during encryption and key generation. Notably, our construction does not rely on the sequential matrix multiplication technique of [GV15] as the IBE scheme of [Yam17]. Finally, we note that the size of the public matrices and ciphertexts are unchanged.

1.2 Related Works

The idea of encoding predicates to another form has already been implicitly or explicitly used in other works. The notion of randomized encoding [IK00, AIK04] (not specific to predicates) aims to trade the computation of a "complex" function $f(x)$ for the computation of a "simpler" randomized function $\hat{f}(x;r)$ whose output distribution on an input x encodes the value for $f(x)$. The notion of predicate encoding [Wee14, CGW15] (and also the related notion of pair encoding

[Att14, Att16]) has already been used previously, in a completely different context, as a generic framework that abstracts the concept of dual system encryption techniques for bilinear maps, and not as a tool for lowering the circuit complexity of predicates.

2 Technical Overview

We now give a brief overview of our technical approaches. A formal treatment is given in the main body. We break our overview in two pieces. First, we give intuition for our notion of predicate encoding schemes PES and illustrate the significance of the MultD-Eq predicates. Then, we overview how the different types of PES schemes for the MultD-Eq predicates can be used to construct VRFs, and PE schemes for the MultD-Eq predicates.

Different Ways of Encoding Predicates. Predicates are often times implicit in cryptographic constructions and in some cases there lies an untapped potential. To highlight this, we recall the observation of [Yam17]. An admissible hash function is one of the central tools used to prove adaptive security (e.g., digital signatures, identity-based encryptions, verifiable random functions). At a high level, during the security proof, it allows the simulator to secretly partition the input space into two disjoint sets, so there is a noticeable probability that the input values submitted by the adversary as challenge queries fall inside the intended sets. Traditionally, the partition made by the admissible hash function is viewed as a bit-fixing predicate; a bit-fixing predicate is specified by a string $K \in \{0,1,\bot\}^\ell$ where the number of non-$\bot$ symbols are $O(\log \lambda)$, and the input space $\{0,1\}^\ell$ is partitioned by the rule whether the string $x \in \{0,1\}^\ell$ matches the string K on all non-$\bot$ symbols.

[Yam17] observed that a bit-fixing predicate can be encoded as a subset predicate; an observation not made since the classical works of [BB04b, CHKP10]. In particular, Yamada observed that K has many meaningless $\bot$ symbols and only has $O(\log \lambda)$ meaningful non-$\bot$ symbols. Under this observation, he managed to encode K into a very small set T_K (e.g., $|\mathsf{T}_K| = O(\log^2 \ell)$) where each element indicates the position of the non-$\bot$ symbols. Now, the partition of the input space is done by checking whether the input includes the set T_K or not. Since admissible hash functions are implicitly embedded in the public parameters, this idea allowed them to significantly reduce the number of public parameters for identity-based encryption (IBE) schemes and the size of the verification key (or the proof size) for VRFs.

We take this observation one step further. A predicate defines a function, but often a function may be represented as a polynomial[7] in various ways depending on what kind of properties we require. This is easiest to explain through an example. Let us continue with the above example of the subset predicate used in [Yam17]: $P_{\mathsf{T}} : 2^{[2n]} \to \{0,1\}$, where $P_{\mathsf{T}}(\mathsf{S}) = 1$ iff $\mathsf{T} \subseteq \mathsf{S}$. Here, assume $|\mathsf{T}| = m$ and all the inputs to P_{T} have cardinality n. One of the most natural ways to represent the subset predicate as a polynomial is by its boolean circuit representation:

$$\underbrace{\prod_{i=1}^{m}\Big(1-\prod_{j=1}^{n}\Big(1-\underbrace{\prod_{k=1}^{\zeta}\Big(1-(t_{i,k}-s_{j,k})^2\Big)}_{\text{is } t_i=s_j?}\Big)\Big)}_{\text{is } t_i\in\mathsf{S}?} = \begin{cases} 1 & \text{if} \quad \mathsf{T}\subseteq\mathsf{S} \\ 0 & \text{if} \quad \mathsf{T}\not\subseteq\mathsf{S} \end{cases}, \tag{1}$$

where $\zeta = \lfloor \log 2n \rfloor + 1$, $\mathsf{T} = \{t_i\}_{i\in[m]}, \mathsf{S} = \{s_j\}_{j\in[n]} \subseteq [2n]$ and $t_{i,k}, s_{j,k}$ are the k-th bit of the binary representation of t_i, s_j. Here Eq. (1) is the polynomial representation of the boolean

[7] It might be more precise to state that a predicate is represented by a circuit, however, in this section we adopt the view of polynomials to better convey the intuition.

logic $\bigwedge_{i\in[m]}\bigvee_{j\in[n]}\bigwedge_{k\in[\zeta]}(t_{i,k} = s_{j,k})$. This is essentially what was used for the lattice-based IBE construction of [Yam17] with very short public parameters. Observe that this polynomial has degree $2mn\zeta$, which is $O(\lambda \log^3 \lambda)$ if we are considering the subset predicate specifying the admissible hash function, where we have $m = O(\log^2 \lambda), n = O(\lambda)$ and $\zeta = O(\log \lambda)$. However, in general, using a high degree polynomial may be undesirable for many reasons, even if it is only of degree linear in the security parameter. For the case of the IBE scheme of [Yam17], due to the highly multiplicative structure, the encryption and key generation algorithms require to rely on a linear number of heavy sequentialized matrix multiplication technique of [GV15]. Therefore, it is a natural question to ask whether we can embed a predicate into a polynomial with lower degree, and in some cases into a linear polynomial.

Indeed, we show that it is possible for the above predicate. Namely, we can do much better by noticing the extra structure of subset predicates; we know there exists at most one $j \in [n]$ that satisfies $t_i = s_j$. Therefore, we can equivalently express Eq. (1) as the following polynomial:

$$\prod_{i=1}^{m}\sum_{j=1}^{n}\prod_{k=1}^{\zeta}\left(1-(t_{i,k}-s_{j,k})^2\right) = \begin{cases} 1 & \text{if} \quad \mathsf{T} \subseteq \mathsf{S} \\ 0 & \text{if} \quad \mathsf{T} \not\subseteq \mathsf{S} \end{cases} . \tag{2}$$

This polynomial is now down to degree $2m\zeta$. When this subset predicate specifies the admissible hash function, Eq. (2) significantly lowers the degree down to $O(\log^3 \lambda)$. Furthermore, if we do not require the output to be exactly 0 or 1, and only care that the predicate behaves differently on satisfied/non-satisfied inputs, we can further lower the degree down to 2ζ. In particular, consider the following polynomial:

$$m - \sum_{i=1}^{m}\sum_{j=1}^{n}\prod_{k=1}^{\zeta}\left(1-(t_{i,k}-s_{j,k})^2\right) = \begin{cases} 0 & \text{if} \quad \mathsf{T} \subseteq \mathsf{S} \\ \neq 0 & \text{if} \quad \mathsf{T} \not\subseteq \mathsf{S} \end{cases} , \tag{3}$$

which follows from the observation that $|\mathsf{T}| = m$. Since, the output of the polynomial is different for the case $\mathsf{T} \subseteq \mathsf{S}$ and $\mathsf{T} \not\subseteq \mathsf{S}$, Eq. (3) indeed properly encodes the information of the subset predicate. Using this polynomial instead of Eq. (1) already allows us to significantly optimize the concrete parameters of the lattice-based IBE of [Yam17]. In fact, by encoding the inputs T, S in a different way and with some additional ideas, we can encode the subset predicate into a *linear* polynomial.

To summarize, depending on what we require for the encoding of a predicate (e.g., preserve the functionality, linearize the encoding) one has the freedom of choosing how to express a particular predicate. We formalize this idea of a "right encoding" by introducing the notion of *predicate encoding schemes*. In the above we used the subset predicate as an motivating example, however, in our work we focus on a wider class of predicates called the *multi-dimensional equality* MultD-Eq predicates, and propose two encoding schemes $\mathsf{PES}_{\mathsf{FP}}$ and $\mathsf{PES}_{\mathsf{Lin}}$ with different applications in mind.

Finally, we state two justifications for why we pursue the construction of predicate encoding schemes for the class of MultD-Eq predicates. First, the MultD-Eq predicates are expressive enough to encode many useful predicates that come up in cryptography (e.g., bit-fixing, subset conjunction, range conjunction predicates), that being for constructions of cryptographic primitives or for embedding secret information during in the security proof. Second, in spite of its expressiveness, the MultD-Eq predicates have a simple structure that we can exploit and offers us plenty of freedom on the types of predicate encoding schemes we can achieve. The definition and a more detailed discussion on the expressiveness of MultD-Eq are provided in Sec. 4.2, 4.3 and Appendix B.1.

Constructing VRFs. Similarly to many of the prior works [BMR10, ACF14, Jag15, Yam17] on VRFs with all the desired properties, we use admissible hash functions and base security on the L-DDH assumption, which states that given $(h, g, g^{\alpha}, \cdots, g^{\alpha^L}, \Psi)$ it is hard to distinguish whether $\Psi = e(g,h)^{1/\alpha}$ or a random element. Here, we briefly review the core idea used during the security proof of [Yam17] for the pseudorandomness property of the VRF. We note that many of the arguments made below are informal for the sake of intuition. Their observation was that the admissible hash function embedded during simulation can be stated in the following way using a subset predicate:

$$\mathsf{F}_{\mathsf{T}}(X) = \begin{cases} 0 & \text{if } \mathsf{T} \subseteq \mathsf{S}(X) \\ 1 & \text{if } \mathsf{T} \not\subseteq \mathsf{S}(X) \end{cases} \qquad \text{where} \quad \mathsf{S}(X) = \{2i - C(X)_i \mid i \in [n]\}.$$

Here, $C(\cdot)$ is a public hash function that maps an input X (of the VRF) to a bit string $\{0,1\}^n$, and $\mathsf{T} \subseteq [2n]$ is a set defined as $\mathsf{T} = \{2i - K_i \mid i \in [n], K_i \neq \perp\}$ where K is the secret string in $\{0,1,\perp\}^n$ that specifies the partition made by the admissible hash. Since, the number of non-$\perp$ symbols in K are $O(\log^2 \lambda)$, the above function can be represented by a set T with cardinality $O(\log^2 \lambda)$. During security proof, by the property and definition of F_{T}, we have

$$\Big(\mathsf{T} \not\subseteq \mathsf{S}(X^{(1)})\Big) \ \wedge \ \cdots \ \wedge \ \Big(\mathsf{T} \not\subseteq \mathsf{S}(X^{(Q)})\Big) \ \wedge \ \Big(\mathsf{T} \subseteq \mathsf{S}(X^*)\Big),$$

with non-negligible probability, where X^* is the challenge input and $X^{(1)}, \cdots, X^{(Q)}$ are the inputs for which the adversary has made evaluation queries. The construction of [Yam17] is based on previous inversion-based VRFs [DY05, BMR10]. Here, we ignore the problem of how to add verifiability to the scheme and overview on how they prove pseudorandomness of the VRF evaluation. Informally, during simulation, the simulator uses the following polynomial to encode the admissible hash function:

$$Q(\alpha) \Big/ \Big(\prod_{i=1}^{m}\prod_{j=1}^{n}(\alpha + t_i - s_j)\Big) = \begin{cases} \frac{\text{const}}{\alpha} + \mathsf{poly}(\alpha) & \text{if} \quad \mathsf{T} \subseteq \mathsf{S}(X) \\ \mathsf{poly}(\alpha) & \text{if} \quad \mathsf{T} \not\subseteq \mathsf{S}(X) \end{cases}, \tag{4}$$

where $Q(\alpha)$ is some fixed polynomial with degree roughly $4n$ independent of the input X. Here, recall $\alpha \in \mathbb{Z}_p$ is that of the L-DDH problem, and notice that in Eq. (4) the polynomial will have α in the denominator if and only if $\mathsf{T} \subseteq \mathsf{S}(X)$. Although this may not seem quite like it, this polynomial is indeed an encoding of the subset predicate[8] since it acts differently depending on $\mathsf{T} \subseteq \mathsf{S}(X)$ and $\mathsf{T} \not\subseteq \mathsf{S}(X)$. Finally, we note that the output Y of the VRF is obtained by simply putting the above polynomial in the exponent of $e(g,h)$.

Now, if the simulator is given enough $(g^{\alpha^i})_{i\in[L]}$ as the L-DDH challenge, it can create a valid evaluation Y for inputs X such that $\mathsf{T} \not\subseteq \mathsf{S}(X)$, since it can compute terms of the form $e(g^{\mathsf{poly}(\alpha)}, h) = e(g,h)^{\mathsf{poly}(\alpha)}$. Furthermore, for the challenge query X^* it will use Ψ; if $\Psi = e(g,h)^{1/\alpha}$ it can correctly simulate for the case $\mathsf{T} \subseteq \mathsf{S}(X^*)$, otherwise the evaluation Y^* of the VRF is independent of X^*. Therefore, under the hardness of the L-DDH assumption, the output is proven pseudorandom. Observe that for the simulator to compute $e(g,h)^{\mathsf{poly}(\alpha)}$ from Eq. (4), it needs to have $(g^{\alpha^i})_{i\in[L]}$ where $L = O(n)$. Then, since $n = O(\lambda)$, we need to base this on an L-DDH assumption where $L = O(\lambda)$.[9] To reflect the above polynomial, the verification keys

[8] To be strict, this does not exactly fit the definition of predicate encoding we define in Sec. 4. However, we can do so by appropriately arguing the size of α or by viewing α as an indeterminate.

[9] In the actual construction we require $L = \omega(\lambda \log \lambda)$, since we need to simulate a higher degree polynomial in the exponent.

are set as $(h, \hat{g}, (W_i = \hat{g}^{w_i}))$ in the actual construction. During simulation the parameters are (roughly) set as $\hat{g} = g^{Q(\alpha)}$, $\hat{g}^{w_i} = \hat{g}^{\alpha + t_i}$.

The above construction is rather naive in that it checks whether $\mathsf{T} \subseteq \mathsf{S}(X)$ in a brute-force manner (as also noted in [Yam17]). Our idea is to instead use the polynomial from Eq. (2) to represent the admissible hash function. In other words, we embed the following polynomial during simulation:

$$\frac{1}{\alpha} \cdot \prod_{i=1}^{m} \sum_{j=1}^{n} \prod_{k=1}^{\zeta} \left(1 - (\alpha + t_{i,k} - s_{j,k})^2\right) = \begin{cases} \frac{1}{\alpha} + \mathsf{poly}(\alpha) & \text{if} \quad \mathsf{T} \subseteq \mathsf{S}(X) \\ \mathsf{poly}(\alpha) & \text{if} \quad \mathsf{T} \not\subseteq \mathsf{S}(X) \end{cases}. \tag{5}$$

We note that in our actual construction, we use an optimized version of Eq. (2) called $\mathsf{PES}_{\mathsf{FP}}$. Similarly to above, we put the above polynomial in the exponent of $e(g, h)$ for the VRF evaluation. The difference is that the degree of the polynomial in Eq. (5) is significantly lowered down to merely $2m\zeta$, which is $O(\log^3 \lambda)$. Therefore, when the simulator needs to compute $e(g, h)^{\mathsf{poly}(\alpha)}$ during simulation, we only require $(g^{\alpha^i})_{i \in [L]}$ for $L = O(\log^3 \lambda)$. Hence, we significantly reduced the required L of the L-DDH assumption to poly-logarithm. Note that we need to validate the output in a different way now, since the terms α, t_i, s_j that appear in the left-hand polynomial are not in the denominator as in Eq. (4). Now, to generate the proof, we take the so called "step ladder approach" [Lys02, ACF09, HW10], where we publish values of the form $(g^{\theta_{i'}})_{i' \in [m]}, (g^{\theta_{i,j,k'}})_{(i,j,k') \in [m] \times [n] \times [\zeta]}$ defined as follows:

$$\theta_{i'} = \prod_{i=1}^{i'} \sum_{j=1}^{n} \prod_{k=1}^{\zeta} \left(1 - (w_{i,k} - s_{j,k})^2\right), \quad \theta_{i,j,k'} = \prod_{k=1}^{k'} \left(1 - (w_{i,k} - s_{j,k})^2\right),$$

where we (roughly) set $g^{w_{i,k}}$ as $g^{\alpha + t_{i,k}}$ during simulation. Although this scheme achieves a very short verification key, it comes at the cost of a rather long proof size of $O(mn\zeta) = O(\lambda \log^3 \lambda)$.

Finally, we describe how to make the proof much shorter, while still maintaining a sub-linear verification key size. As a first step, we can use the simple trick used in [Yam17] to make the proof much shorter. Namely, we add helper components to the verification key so that anyone can compute $(\theta_{i,j,k'})$ publicly. However, as in [Yam17], this leads to a long verification key with size $\tilde{\Omega}(\lambda)$. Interestingly, for our construction, we can do much better and shorten the verification key by a quadratic factor by in a sense *skipping* some ladders. The main observation is the additive structure in $(\theta_{i'})_{i'}$. In particular, if each $\theta_{i'}$ were simply a large product $\prod_{i,j,k} \left(1 - (w_{i,k} - s_{j,k})^2\right)$, we would have to prepare all the necessary helper components in the verification key that would allow to compute $g^{\theta_{i,j,\zeta}}$. This is because in the step ladder approach, after computing $g^{\theta_{i,j,\zeta}}$, we have to reuse this as an input to the bilinear map to validate the next term in the ladder. However, in our case, we only need the ability to publicly compute $e(g, g)^{\theta_{i,j,\zeta}}$. Here, we crucially rely on the additive structure in $\theta_{i'}$ that allows us to compute $e(g, g)^{\sum_{j \in [n]} \theta_{i,j,\zeta}}$ by ourselves; thus the notion of skipping some ladders. Note that we are not able to publicly compute $e(g, g)^{\prod_{j \in [n]} \theta_{i,j,\zeta}}$. Finally, we continue with the step ladder approach for the outer $\prod_{i=1}^{i'}$ products. Therefore, since we only need the ability to generate $e(g, g)^{\theta_{i,j,\zeta}}$ rather than $g^{\theta_{i,j,\zeta}}$, we can reduce quadratically the number of helper components we have to publish in the verification key.

Constructing PE for the MultD-Eq Predicates. Our proposed predicate encryption scheme for the MultD-Eq predicates follows the general framework of [AFV11, BGG+14], which allows us to compute an inner product of a private attribute vector X associated to a ciphertext and a (public) predicate vector Y associated to a secret key. To accommodate this framework, we use our proposed *linear* predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}}$ for the MultD-Eq predicates. In the

overview, we continue with our examples with the subset predicate for simplicity. The core idea is the same as for the MultD-Eq predicates. Essentially, $\mathsf{PES}_{\mathsf{Lin}}$ will allow us to further modify Eq. (3), to the following linear polynomial:

$$\sum_{i=1}^{L} a_i \mathsf{X}_i = \begin{cases} 0 & \text{if} \quad \mathsf{T} \subseteq \mathsf{S} \\ \neq 0 & \text{if} \quad \mathsf{T} \not\subseteq \mathsf{S} \end{cases}, \tag{6}$$

where $(\mathsf{X}_i)_{i\in[L]}, (a_i)_{i\in[L]} \in \mathbb{Z}_q^L$ are encodings of the attribute set T and the predicate set S, respectively.

Following the general framework, the secret key for a user with predicate set S is a short vector $\mathbf{e}$ such that $[\mathbf{A}|\mathbf{B}_{\mathsf{S}}]\mathbf{e} = \mathbf{u}$ for a random public vector $\mathbf{u}$, where $\mathbf{B}_{\mathsf{S}}$ is defined as in Eq. (7) below. Furthermore, we privately embed an attribute set T into the ciphertext as

$$[\mathbf{c}_1^\top \mid \cdots \mid \mathbf{c}_L^\top] = \mathbf{s}^\top [\mathbf{B}_1 + \mathsf{X}_1\mathbf{G} \mid \cdots \mid \mathbf{B}_L + \mathsf{X}_L\mathbf{G}] + [\mathbf{z}_1^\top \mid \cdots \mid \mathbf{z}_L^\top].$$

Using the gadget matrix $\mathbf{G}$ of [MP12], a user corresponding to the predicate set S can transform the ciphertext without knowledge of T as follows:

$$\sum_{i=1}^{L} \mathbf{c}_i^\top \mathbf{G}^{-1}(a_i\mathbf{G}) = \mathbf{s}^\top \Big(\underbrace{\sum_{i=1}^{L} \mathbf{B}_i \mathbf{G}^{-1}(a_i\mathbf{G})}_{=\mathbf{B}_{\mathsf{S}}} + \sum_{i=1}^{L} a_i \mathsf{X}_i \cdot \mathbf{G} \Big) + \underbrace{\sum_{i=1}^{L} \mathbf{z}_i^\top \mathbf{G}^{-1}(a_i\mathbf{G})}_{=\mathbf{z}\ \text{(noise term)}}. \tag{7}$$

Observe the matrix $\mathbf{B}_{\mathsf{S}}$ is defined independently of X (i.e., the attribute set S). By Eq. (6) and the correctness of the predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}}$, we have $\sum_{i\in[L]} a_i\mathsf{X}_i = 0$ when the subset predicate is satisfied, as required for decryption. To prove security, we set the matrices $\{\mathbf{B}_i\}_{i\in[L]}$ as $\mathbf{B}_i = \mathbf{A}\mathbf{R}_i - \mathsf{X}_i^* \cdot \mathbf{G}$, where $\mathbf{A}$ is from the problem instance of LWE, $\mathbf{R}_i$ is a random matrix with small coefficients and $(\mathsf{X}_i^*)_{i\in[L]}$ is the encoding of the challenge attribute set T^*. During simulation we have

$$\mathbf{B}_{\mathsf{S}} = \mathbf{A}\mathbf{R}_{\mathsf{S}} - \sum_{i=1}^{L} a_i \mathsf{X}^* \mathbf{G}, \quad \text{where} \quad \mathbf{R}_{\mathsf{S}} = \sum_{i=1}^{L} \mathbf{R}_i \mathbf{G}^{-1}(a_i\mathbf{G}).$$

for any set S. Here, we have $\sum_{i\in[L]} a_i\mathsf{X}^* \neq 0$ iff $\mathsf{T}^* \not\subseteq \mathsf{S}$. Therefore, for the key extraction queries for S such that $\mathsf{T}^* \not\subseteq \mathsf{S}$, we can use $\mathbf{R}_{\mathsf{S}}$ as the $\mathbf{G}$-trapdoor [MP12] for the matrix $[\mathbf{A}|\mathbf{B}_{\mathsf{S}}]$ to simulate the secret keys. We are able to generate the challenge ciphertext for the subset T^* by computing

$$\underbrace{(\mathbf{s}^\top\mathbf{A} + \mathbf{z}'^\top)}_{\text{LWE Problem}}[\mathbf{I}|\mathbf{R}_1|\cdots|\mathbf{R}_L] = \mathbf{s}^\top[\mathbf{A}|\mathbf{B}_1 + \mathsf{X}_1^*\mathbf{G}|\cdots|\mathbf{B}_L + \mathsf{X}_L^*\mathbf{G}] + \underbrace{\mathbf{z}'^\top[\mathbf{I}|\mathbf{R}_1|\cdots|\mathbf{R}_L]}_{\text{simulation noise term}}$$

A subtle point here is that the simulation noise term is not distributed correctly as in Eq. (7). However, this can be resolved by the noise rerandomization technique of [KY16].[10]

Finally, we propose a technique to finer analyze the growth of the noise term $\mathbf{z} = \sum_{i\in[L]} \mathbf{z}_i^\top \mathbf{G}^{-1}(a_i\mathbf{G})$ and the $\mathbf{G}$-trapdoor $\mathbf{R}_{\mathsf{S}} = \sum_{i\in[L]} \mathbf{R}_i\mathbf{G}^{-1}(a_i\mathbf{G})$ used during simulation. This allows us to choose narrower Gaussian parameters and let us base security on a weaker LWE assumption. The main observation is that $\mathbf{G}^{-1}(a_i\mathbf{G}) \in \{0,1\}^{nk\times nk}$ is a block-diagonal matrix with n square matrices with size k along its diagonals where $n = O(\lambda)$ and $k = O(\log\lambda)$. Exploiting this additional block-diagonal structure, we are able to finer control the growth of $\|\mathbf{v}\|_2$ and $s_1(\mathbf{R}_{\mathsf{S}})$ (i.e., the largest singular value of $\mathbf{R}_{\mathsf{S}}$).

[10] Alternatively, we could have used the techniques of [AFV11, BGG+14] and altered the real scheme by multiplying the error vectors of the ciphertexts by random matrices with small coefficients.

3 Preliminaries

Notations. We use $\{\cdot\}$ to denote sets and use $(\cdot)$ to denote a finite ordered list of elements. When we use notations such as $(w_{i,j})_{(i,j)\in[n]\times[m]}$ for $n, m \in \mathbb{N}$, we assume the elements are sorted in the lexicographical order. For $n, m \in \mathbb{N}$ with $n \leq m$, denote $[n]$ as the set $\{1, \cdots, n\}$ and $[n, m]$ as the set $\{n, \cdots, m-1, m\}$. For a vector $\mathbf{v} \in \mathbb{R}^n$, denote $\|\mathbf{v}\|_2$ as the Euclidean norm. For a matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$, denote $\|\mathbf{R}\|_{\mathsf{GS}}$ as the longest column of the Gram-Schmidt orthogonalization of $\mathbf{R}$ and denote $s_1(\mathbf{R})$ as the largest singular value.

3.1 Verifiable Random Functions

We define a verifiable random function $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$ as a tuple of three probabilistic polynomial time algorithms [MRV99].

$\mathsf{Gen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$: The key generation algorithm takes as input the security parameter 1^λ and outputs a verification key vk and a secret key sk.

$\mathsf{Eval}(\mathsf{sk}, X) \to (Y, \pi)$: The evaluation algorithm takes as input the secret key sk and an input $X \in \{0,1\}^n$, and outputs a value $Y \in \mathcal{Y}$ and a proof π, where $\mathcal{Y}$ is some finite set.

$\mathsf{Verify}(\mathsf{vk}, X, (Y, \pi)) \to 0/1$: The verification algorithm takes as input the verification key vk, $X \in \{0,1\}^n$, $Y \in \mathcal{Y}$ and a proof π, and outputs a bit.

Definition 1. *We say a tuple of polynomial time algorithms* $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$ *is a verifiable random function if all of the following requirements hold:*

Correctness. *For all* $\lambda \in \mathbb{N}$*, all* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and all* $X \in \{0,1\}^n$*, if* $(Y, \pi) \leftarrow \mathsf{Eval}(\mathsf{sk}, X)$ *then* $\mathsf{Verify}(\mathsf{vk}, X, (Y, \pi))$.

Uniqueness. *For an arbitrary string* $\mathsf{vk} \in \{0,1\}^*$ *(not necessarily generated by* Gen*) and all* $X \in \{0,1\}^n$*, there exists at most a single* $Y \in \mathcal{Y}$ *for which there exists an accepting proof* π*.*

Pseudorandomness. *This security notion is defined by the following game between a challenger and an adversary* $\mathcal{A}$*.*

> **Setup.** *The challenger runs* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and gives* vk *to* $\mathcal{A}$*.*
>
> **Phase 1.** $\mathcal{A}$ *adaptively submits an evaluation query* $X \in \{0,1\}^n$ *to the challenger, and the challenger returns* $(Y, \pi) \leftarrow \mathsf{Eval}(\mathsf{sk}, X)$*.*
>
> **Challenge Query.** *At any point,* $\mathcal{A}$ *may submit a challenge input* $X^* \in \{0,1\}^n$*. Here, we require that* $\mathcal{A}$ *has not submitted* X^* *as an evaluation query in Phase 1. The challenger picks a random coin* $\mathsf{coin} \leftarrow \{0,1\}$*. Then it runs* $(Y_0^*, \pi_0^*) \leftarrow \mathsf{Eval}(\mathsf{sk}, X^*)$ *and picks* $Y_1^* \leftarrow \mathcal{Y}$*. Finally it returns* Y_{coin}^* *to* $\mathcal{A}$*.*
>
> **Phase 2.** $\mathcal{A}$ *may continue on submitting evaluation queries as in Phase 1 with the added restriction that* $X \neq X^*$*.*
>
> **Guess.** *Finally,* $\mathcal{A}$ *outputs a guess* $\widehat{\mathsf{coin}}$ *for* coin*.*

The advantage of $\mathcal{A}$ *is defined as* $|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}|$*. We say that the* VRF *satisfies (adaptive) pseudorandomness if the advantage of any probabilistic polynomial time algorithm* $\mathcal{A}$ *is negligible.*

3.2 Predicate Encryptions

We present the definition of predicate encryption [BW07, KSW08, AFV11]. A predicate encryption PE scheme with attribute space $\mathcal{X}$ and predicate space $\mathcal{P}$ consists of four probabilistic polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm takes as input a security parameter 1^λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, P) \to \mathsf{sk}_P$: The key generation algorithm takes as input the master public key mpk, the master secret key msk, and a predicate $P \in \mathcal{P}$. It outputs a secret key sk_P. We assume the description of P is implicitly included in sk_P.

$\mathsf{Encrypt}(\mathsf{mpk}, X, \mathsf{M}) \to \mathsf{ct}$: The encryption algorithm takes as input a master public key mpk, an attribute vector $X \in \mathcal{X}$ and a message M. It outputs a ciphertext ct.

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_P, \mathsf{ct}) \to \mathsf{M}$ or $\perp$: The decryption algorithm takes as input the master public key mpk, a secret key sk_P, and a ciphertext ct. It outputs the message M or $\perp$, which means that the ciphertext is not in a valid form.

Definition 2. *We say a tuple of algorithms* $\mathsf{PE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a predicate encryption scheme if all of the following requirements hold:*

Correctness. *For all* $\lambda \in \mathbb{N}$*, all* $X \in \mathcal{X}, P \in \mathcal{P}$ *such that* $P(X) = 1$[11] *and all* M *in the specified message space,* $\Pr[\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_P, \mathsf{Encrypt}(\mathsf{mpk}, X, \mathsf{M})) = \mathsf{M}] = 1 - \mathsf{negl}(\lambda)$ *holds, where the probability is taken over the randomness used in all of the algorithms.*

Security. *This security notion is defined by the following game between a challenger and an adversary* $\mathcal{A}$*.*

> **Setup.** *At the outset of the game,* $\mathcal{A}$ *submits to the challenger an attribute* $X^* \in \mathcal{X}$ *on which it wishes to be challenged. Then, the challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives the public parameter* mpk *to* $\mathcal{A}$*.*
>
> **Phase 1.** $\mathcal{A}$ *adaptively submits key extraction queries. If* $\mathcal{A}$ *submits a predicate* $P \in \mathcal{P}$ *to the challenger, the challenger returns* $\mathsf{sk}_P \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, P)$*. Here, we require the predicates* P *to satisfy* $P(X^*) = 0$ *(that is,* sk_P *does not decrypt the challenge ciphertext).*
>
> **Challenge Phase.** *At any point,* $\mathcal{A}$ *outputs a message* M^**. The challenger picks a random coin* $\mathsf{coin} \leftarrow \{0, 1\}$ *and a random ciphertext* ct_1^* *from the ciphertext space. If* $\mathsf{coin} = 0$*, it runs* $\mathsf{ct}_0^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, X^*, \mathsf{M}^*)$ *and gives the challenge ciphertext* ct_0^* *to* $\mathcal{A}$*. If* $\mathsf{coin} = 1$*, it gives* ct_1^* *to* $\mathcal{A}$*.*
>
> **Phase 2.** $\mathcal{A}$ *may continue to make key extraction queries as in Phase 1.*
>
> **Guess.** *Finally,* $\mathcal{A}$ *outputs a guess* $\widehat{\mathsf{coin}}$ *for* coin*.*

The advantage of $\mathcal{A}$ *is defined as* $|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}|$*. We say that the* PE *scheme is* selectively secure *and* weakly attribute hiding*, if the advantage of any PPT* $\mathcal{A}$ *is negligible.*

[11] We follow the convention that $P(X) = 1$ signifies the ability to decrypt. This is opposite to the convention used in the recent lattice-based schemes, and is done purely for convenience of our presentation.

3.3 Background on Lattices

A (full-rank-integer) m-dimensional lattice Λ in $\mathbb{Z}^m$ is a set of the form $\{\sum_{i\in[m]} x_i\mathbf{b}_i | x_i \in \mathbb{Z}\}$, where $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_m\}$ are m linearly independent vectors in $\mathbb{Z}^m$. We call $\mathbf{B}$ the basis of the lattice Λ. For any positive integers n, m and $q \geq 2$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m | \mathbf{Az} = \mathbf{0} \mod q\}$, and $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m | \mathbf{Az} = \mathbf{u} \mod q\}$.

Gaussian Measures. For an m-dimensional lattice Λ, the discrete Gaussian distribution over Λ with center $\mathbf{c}$ and parameter σ is defined as $D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \rho_{\sigma,\mathbf{c}}(\mathbf{x})/\rho_{\sigma,\mathbf{c}}(\Lambda)$ for all $\mathbf{x} \in \Lambda$, where $\rho_{\sigma,\mathbf{c}}(\mathbf{x})$ is a Gaussian function defined as $\exp(-\pi\|\mathbf{x}-\mathbf{c}\|^2/\sigma^2)$ and $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. Further for an m-dimensional shifted lattice $\Lambda + \mathbf{t}$, we define the Gaussian distribution $D_{\Lambda+\mathbf{t},\sigma}$ with parameter σ as the process of adding the vector $\mathbf{t}$ to a sample from $D_{\Lambda,\sigma,-\mathbf{t}}$. Finally, we call D a B-bounded distribution, if all the elements in the support of D have absolute value smaller than B.

Hardness Assumption. We define the Learning with Errors (LWE) problem introduced by Regev [Reg05].

Definition 3 (Learning with Errors)**.** *For integers $n = n(\lambda), m = m(n)$, a prime $q = q(n) > 2$, an error distribution over $\chi = \chi(n)$ over $\mathbb{Z}$, and a PPT algorithm $\mathcal{A}$, an advantage for the* learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ *of $\mathcal{A}$ is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}} = \left| \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{A}^{\top}\mathbf{s} + \mathbf{z}) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{w} + \mathbf{z}) = 1\right] \right|$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{w} \leftarrow \mathbb{Z}_q^m$, $\mathbf{z} \leftarrow \chi$. We say that the LWE *assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}}$ is negligible for all PPT $\mathcal{A}$.*

The (decisional) $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$ for $\alpha q > 2\sqrt{n}$ has been shown by Regev [Reg05] to be as hard as approximating the worst-case SIVP and GapSVP problems to within $\tilde{O}(n/\alpha)$ factors in the ℓ_2-norm in the worst case. In the subsequent works, (partial) dequantumization of the reduction were achieved [Pei09, BLP+13].

Random Matrices. The following lemmas state the properties of random matrices. They will be used to obtain a more precise analysis of our lattice-based scheme.

Lemma 1 ([LPRTJ05, ABB10])**.** *Let m, k be positive integers such that $k \geq m$. If $\mathbf{R}$ is sampled uniformly in $\{-1, 1\}^{m\times k}$ then $s_1(\mathbf{R}) \leq 20\sqrt{m+k}$ with overwhelming probability in m.*

The proof for the following lemma appears in Appendix A.

Lemma 2. *Let ℓ, n, k be positive integers and set $m = nk$, and let D be a B-bounded distribution. Let $\mathbf{R} \leftarrow D^{\ell\times m}$ and $\mathbf{U}$ be an arbitrary block diagonal matrix $\mathbf{U} = \mathrm{diag}(\mathbf{U}^{(1)}, \cdots \mathbf{U}^{(n)}) \in \{0, 1\}^{m\times m}$ where $\mathbf{U}^{(w)} \in \{0, 1\}^{k\times k}$ for $w \in [n]$. Then, there exists a universal constant $C > 0$ such that we have $s_1(\mathbf{RU}) \leq C \cdot Bm\sqrt{k} = C \cdot Bnk^{3/2}$ with all but negligible probability in m.*

Lemma 3 (Leftover Hash Lemma)**.** *Let $q > 2$ be a prime, m, n, k be positive integers such that $m > (n + 1)\log q + \omega(\log n)$, $k = \mathsf{poly}(n)$ and let $\mathbf{R} \leftarrow \{-1, 1\}^{m\times k}$. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$ respectively. Then the distribution of $(\mathbf{A}, \mathbf{AR})$ is negligibly close in n to the distribution of $(\mathbf{A}, \mathbf{B})$.*

Discrete Gaussian Lemmas. The following lemmas are used to manipulate and obtain meaningful bounds on discrete Gaussian vectors.

Lemma 4 ([ABB10], Lem. 8)**.** *Let n, m, q be positive integers with $m > n$, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be a matrix, $\mathbf{u} \in \mathbb{Z}_q^n$ be a vector, $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ be a basis for $\Lambda^\perp(\mathbf{A})$, and $\sigma > \|\mathbf{T_A}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$. Then, if we sample a vector $\mathbf{x} \leftarrow D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}),\sigma}$, we have $\Pr[\|\mathbf{x}\|_2 > \sqrt{m}\sigma] < \mathsf{negl}(n)$.*

Lemma 5 (Noise Rerandomization, [KY16], Lem. 1)**.** *Let q, ℓ, m be positive integers and r a positive real satisfying $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log \ell})\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{z}$ chosen from $D_{\mathbb{Z}^m,r}$. Then for any $\mathbf{V} \in \mathbb{Z}^{m\times\ell}$ and positive real $\sigma > s_1(\mathbf{V})$, there exists a PPT algorithm $\mathsf{ReRand}(\mathbf{V}, \mathbf{b}+\mathbf{z}, r, \sigma)$ that outputs $\mathbf{b}'^\top = \mathbf{b}^\top\mathbf{V} + \mathbf{z}'^\top \in \mathbb{Z}_q^\ell$ where $\mathbf{z}'$ is distributed statistically close to $D_{\mathbb{Z}^\ell,2r\sigma}$.*

Gadget Matrix. We use the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ defined in [MP12]. Without loss of generality, we will always assume that $n|m$. Here, $\mathbf{G}$ is a full rank matrix such that the lattice $\Lambda^\perp(\mathbf{G})$ has a publicly known basis $\mathbf{T_G}$ with $\|\mathbf{T_G}\|_{\mathsf{GS}} \le \sqrt{5}$. With an abuse of notation, we also define a deterministic polynomial time algorithm $\mathbf{G}^{-1}$ that given an input $\mathbf{U} \in \mathbb{Z}_q^{n\times m}$ outputs a matrix $\mathbf{V} \in \{0,1\}^{m\times m}$ such that $\mathbf{GV} = \mathbf{U} \mod q$. In particular, for any $t \in \mathbb{Z}_q$, $\mathbf{G}^{-1}(t \cdot \mathbf{G})$ returns a block diagonal matrix with n square matrices with size m/n along its diagonals.

Sampling Algorithms. The following lemma states useful algorithms for sampling short vectors from lattices.

Lemma 6. ([GPV08, ABB10, CHKP10, MP12]) *Let $n, m, q > 0$ be integers with $m > 2n\lceil\log q\rceil$.*

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, \mathbf{T_A})$*: There exists a randomized algorithm that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a full-rank matrix $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, where $\mathbf{T_A}$ is a basis for $\Lambda^\perp(\mathbf{A})$, $\mathbf{A}$ is statistically close to uniform and $\|\mathbf{T_A}\|_{\mathsf{GS}} = O(\sqrt{n\log q})$.*
- $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{u}, \mathbf{T_A}, \sigma) \to \mathbf{e}$ *: There exists a randomized algorithm that, given matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, a basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ for $\Lambda^\perp(\mathbf{A})$, and a Gaussian parameter $\sigma > \|\mathbf{T_A}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$, outputs a vector $\mathbf{e} \in \mathbb{Z}^{2m}$ sampled from a distribution which is $\mathsf{negl}(n)$-close to $D_{\Lambda^\perp_{\mathbf{u}}([\mathbf{A}|\mathbf{B}]),\sigma}$.*
- $\mathsf{SampleRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}, t, \mathbf{u}, \mathbf{T_G}, \sigma) \to \mathbf{e}$*: There exists a randomized algorithm that, given a full-rank matrix $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n\times m}$, an invertible element $t \in \mathbb{Z}_q$, a matrix $\mathbf{R} \in \mathbb{Z}^{m\times m}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, a basis $\mathbf{T_G}$ for $\Lambda^\perp(\mathbf{G})$, and a Gaussian parameter $\sigma > s_1(\mathbf{R}) \cdot \|\mathbf{T_G}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$, outputs a vector $\mathbf{e} \in \mathbb{Z}^{2m}$ sampled from a distribution which is $\mathsf{negl}(n)$-close to $D_{\Lambda^\perp_{\mathbf{u}}([\mathbf{A}|\mathbf{AR}+t\mathbf{G}]),\sigma}$.*

3.4 Background on Bilinear Maps.

Certified Group Generators. We define certified bilinear group generators as introduced in [HJ16]. We require that there is an efficient bilinear group generator algorithm GrpGen that on input 1^λ outputs a description of bilinear groups $\mathbb{G}$, $\mathbb{G}_T$ with prime order p and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We also require that GrpGen is certified. Namely, there is an efficient algorithm GrpVfy that on input a (possibly incorrectly generated) description of the bilinear groups and outputs whether the description is valid or not. Furthermore, we require that each group element has unique encoding, which can be efficiently recognized.

Definition 4. *A bilinear group generator is a PPT algorithm GrpGen that takes as input a security parameter 1^λ and outputs $\Pi = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1))$ such that the following requirements are satisfied.*

1. *p is prime and* $\log(p) = \Omega(\lambda)$.
2. *$\mathbb{G}$ and $\mathbb{G}_T$ are subsets of* $\{0,1\}^*$*, defined by the algorithmic descriptions of maps* $\phi : \mathbb{Z}_p \to \mathbb{G}$ *and* $\phi_T : \mathbb{Z}_p \to \mathbb{G}_T$.
3. *$\circ$ and $\circ_T$ are algorithmic descriptions of efficiently computable (in the security parameter) maps* $\circ : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *and* $\circ_T : \mathbb{G}_T \times \mathbb{G}_T \to \mathbb{G}_T$*, such that*
   - $(\mathbb{G}, \circ)$ *and* $(\mathbb{G}_T, \circ_T)$ *form algebraic groups*
   - ϕ *is a group isomorphism from* $(\mathbb{Z}_p, +)$ *to* $(\mathbb{G}, \circ)$
   - ϕ_T *is a group isomorphism from* $(\mathbb{Z}_p, +)$ *to* $(\mathbb{G}_T, \circ_T)$
4. *e is an algorithmic description of an efficiently computable (in the security parameter) bilinear map* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$*. We require that e is non-degenerate, that is,*

$$x \neq 0 \quad \Rightarrow \quad e(\phi(x), \phi(x)) \neq \phi_T(0).$$

Definition 5. *We say that a group generator* GrpGen *is certified, if there exists a deterministic polynomial time algorithm* GrpVfy *with the following properties.*

1. ***Parameter validation.*** *Given a string* Π *(which is not necessarily generated by* GrpGen*), algorithm* $\mathsf{GrpVfy}(\Pi)$ *outputs 1 if and only if* Π *has the form*

$$\Pi = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1))$$

   and all requirements from Definition 4 are satisfied.

2. ***Recognition and unique representation of elements of*** $\mathbb{G}$**.** *Each element in* $\mathbb{G}$ *has a unique representation that is efficiently recognizable. Namely, on input two strings* Π *and* s*,* $\mathsf{GrpVfy}(\Pi, s)$ *outputs 1 if and only if* $\mathsf{GrpVfy}(\Pi) = 1$ *and it holds that* $s = \phi(x)$ *for some* $x \in \mathbb{Z}_p$*. Here* $\phi : \mathbb{Z}_p \to \mathbb{G}$ *denotes the fixed group isomorphism contained in* Π *to specify the representation of elements of* $\mathbb{G}$ *(see Definition 4).*

Hardness Assumption.

Definition 6 (L-Diffie-Hellman Assumption)**.** *For a PPT algorithm* $\mathcal{A}$*, an advantage for the decisional L-Diffie-Hellman problem L-DDH of* $\mathcal{A}$ *with respect to* GrpGen *is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{L\text{-}DDH} = |\Pr[\mathcal{A}(\Pi, g, h, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^L}, \Psi_0) \to 1] - \Pr[\mathcal{A}(\Pi, g, h, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^L}, \Psi_1) \to 1]|,$$

where $\Pi \leftarrow \mathsf{GrpGen}(1^{\lambda}), \alpha \leftarrow \mathbb{Z}_p^*, g, h \leftarrow \mathbb{G}, \Psi_0 = e(g,h)^{1/\alpha}$ *and* $\Psi_1 \leftarrow \mathbb{G}_T$*. We say that L-DDH assumption holds if* $\mathsf{Adv}_{\mathcal{A}}^{L\text{-}DDH}$ *is negligible for all PPT* $\mathcal{A}$.

3.5 Other Facts.

The following lemma is taken from [KY16], and is implicit in [BR09, Jag15, Yam16].

Lemma 7 ([KY16], Lem. 8)**.** *Let us consider a VRF and an adversary* $\mathcal{A}$ *that breaks pseudorandomness with advantage* ϵ*. Let the input space be* $\mathcal{X}$ *and consider a map* γ *that maps a sequence of inputs to a value in* $[0,1]$*. We consider the following experiment. We first execute the security game for* $\mathcal{A}$*. Let* X^* *be the challenge input and* $X_1, \cdots, X_Q$ *be the inputs for which*

evaluation queries were made. We denote $\mathbb{X} = (X^*, X_1, \cdots, X_Q)$*. At the end of the game, we set* $\mathsf{coin}' \in \{0,1\}$ *as* $\mathsf{coin}' = \widehat{\mathsf{coin}}$ *with probability* $\gamma(\mathbb{X})$ *and* $\mathsf{coin}' \leftarrow \{0,1\}$ *with probability* $1 - \gamma(\mathbb{X})$*. Then, the following holds.*

$$\left|\Pr[\mathsf{coin}' = \mathsf{coin}] - \frac{1}{2}\right| \geq \gamma_{\min} \cdot \epsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2}$$

where $\gamma_{\min}$ *(resp.* $\gamma_{\max}$*) is the maximum (resp. minimum) of* $\gamma(\mathbb{X})$ *taken over all possible* $\mathbb{X}$*.*

As noted in [Yam17], the lemma was originally proven for IBE schemes in [KY16], however, the exact same proof works for VRFs.

4 Encoding Predicates with Arithmetic Circuits

Here, we formalize the intuition outlined in the introduction on how to encode predicates as circuits. In doing so, we first define *predicates* and *arithmetic circuits*. Notably, to capture the algebraic properties of circuits, we adapt the view of treating circuits as polynomials and vice versa. (For further details, see [SY10].)

Predicates. A *predicate* is simply a function $P : \mathcal{X} \to \{0,1\}$ over some domain $\mathcal{X}$ with image $\{0,1\}$. In particular, predicate P divides the input space $\mathcal{X}$ into two disjoint sets according to some specified relation. Often times, it will be more meaningful to consider a set of predicates $\mathcal{P} = \{P | P : \mathcal{X} \to \{0,1\}\}$.

Arithmetic Circuits. An *arithmetic circuit* C over a ring $\mathcal{R}$ and a set of variables $X = \{x_1, \cdots, x_n\}$ is a directed acyclic graph as follows, where the vertices of C are called gates. Every gate in C of in-degree 0 (*input gate*) is labelled by either a variable from X or a ring element in $\mathcal{R}$. Every other gate in C is labeled by either $+$ (*addition gate*) of $\times$ (*product gate*) and has in-degree $\geq$2. The unique gate of out-degree 0 is called an *output gate*.[12] The *depth* of C is the length of the longest directed path reaching to the output gate. For two gates u and v in C, if (u, v) is an edge in C, then u is called a child of v, and v is called a parent of u. For a gate v in C, define C_v to be the sub-circuit of C rooted at v.

An arithmetic circuit computes a polynomial in a natural way. For a gate v in C, define $p_v \in \mathcal{R}[X]$ to be the polynomial computed by C_v as follows: If v is an input gate labelled by $\alpha \in \mathcal{R} \cup X$, then $p_v = \alpha$. If v is an addition gate with $v_1, v_2, \cdots, v_k$ as children, then $p_v = \sum_{i \in [k]} p_{v_k}$. If v is a product gate with $v_1, v_2, \cdots, v_k$ as children, then $p_v = \prod_{i \in [k]} p_{v_k}$. For a polynomial $p \in \mathcal{R}[X]$, and a gate v in C, we say that v computes p if $p = C_v$. In particular, we say p is a *polynomial representation* of C when the output gate of C computes p. We define the *degree* of C to be the degree of the maximal-degree monomial of the polynomial representation of C.

Finally, it is clear that given some representation of a polynomial, we can uniquely reconstruct the original arithmetic circuit by iteratively converting each monomials into gates beginning from the most inner monomials and moving outward. Note that since one function may be expressed as a polynomial in number of ways, the reconstructed arithmetic circuit may be different even if it has the same functionality, e.g., although $(x_1 + x_2)^2$ and $x_1^2 + 2x_1x_2 + x_2^2$ have the same functionality, $(x_1 + x_2)^2$ will be of depth 2 consisting of 1 addition gate and 1 product gate, but $x_1^2 + 2x_1x_2 + x_2^2$ will be of depth 2 consisting of 1 addition gate and 3 product gates. In the following work, we will use the terms circuits and polynomials interchangeably.

[12] Here, we only consider arithmetic circuits with a single output.

4.1 Predicate Encoding Scheme

We formalize our main tool: predicate encoding scheme.

Definition 7 (Predicate Encoding Scheme). *Let $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of set of efficiently computable predicates where $\mathcal{P}_\lambda$ is a set of predicates of the form $P : \mathcal{X}_\lambda \to \{0,1\}$ for some input space $\mathcal{X}_\lambda$, and let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of rings. We define a* predicate encoding scheme over a family of rings $\mathcal{R}$ for a family of set of predicates $\mathcal{P}$, *as a tuple of* deterministic *polynomial time algorithms* $\mathsf{PES} = (\mathsf{EncInpt}, \mathsf{EncPred})$ *such that*

- $\mathsf{EncInpt}(1^\lambda, \boldsymbol{x}) \to \hat{\boldsymbol{x}}$: *The input encoding algorithm takes as inputs the security parameter 1^λ and input $\boldsymbol{x} \in \mathcal{X}_\lambda$, and outputs an encoding $\hat{\boldsymbol{x}} \in \{0_{\mathcal{R}_\lambda}, 1_{\mathcal{R}_\lambda}\}^t \subseteq \mathcal{R}_\lambda^t$, where $t = t(\lambda)$ is an integer valued polynomial and $0_{\mathcal{R}_\lambda}, 1_{\mathcal{R}_\lambda}$ denote the zero and identity element of the ring $\mathcal{R}_\lambda$, respectively.*
- $\mathsf{EncPred}(1^\lambda, P) \to \hat{C}$: *The predicate encoding algorithm takes as inputs the security parameter 1^λ and a predicate $P \in \mathcal{P}_\lambda$, and outputs a polynomial representation of an arithmetic circuit $\hat{C} : \mathcal{R}_\lambda^t \to \mathcal{R}_\lambda$. We denote $\hat{\mathcal{C}}_\lambda$ as the set of arithmetic circuits $\{\hat{C} \mid \hat{C} \leftarrow \mathsf{EncPred}(1^\lambda, P), \forall P \in \mathcal{P}_\lambda\}$.*

Correctness. *We require a predicate encoding scheme over a family of rings $\mathcal{R}$ for a family of set of predicates $\mathcal{P}$ to satisfy the following: for all $\lambda \in \mathbb{N}$ and all $b \in \{0,1\}$, there exist disjoint subsets $S_{\lambda,0}, S_{\lambda,1} \subset \mathcal{R}_\lambda$ (i.e., $S_{\lambda,0} \cap S_{\lambda,1} = \phi$), such that for all predicates $P \in \mathcal{P}_\lambda$, all inputs $\boldsymbol{x} \in \mathcal{X}_\lambda$ if $P(\boldsymbol{x}) = b$ then $\hat{C}(\hat{\boldsymbol{x}}) \in S_{\lambda,b}$, where $\hat{\boldsymbol{x}} \leftarrow \mathsf{EncInpt}(1^\lambda, \boldsymbol{x}), \hat{C} \leftarrow \mathsf{EncPred}(1^\lambda, P)$.*

Degree. *We say that a predicate encoding scheme PES is of degree $d = d(\lambda)$ if the maximal degree of the circuits in $\hat{\mathcal{C}}_\lambda$ (in their polynomial representation) is d. In case $d = 1$, we say PES is* linear.

In the following, we will be more loose in our use of notations. For simplicity, we omit the subscripts expressing the domain or the security parameter such as $0_{\mathcal{R}}, S_{\lambda,b}, \mathcal{R}_\ell$ when it is clear from context. We also omit the expression family and simply state that it is a predicate encoding scheme over a ring $\mathcal{R}$ for a set of predicates P. Finally, in the following we assume that the algorithms $\mathsf{EncInpt}(1^\lambda, \cdot), \mathsf{EncPred}(1^\lambda, \cdot)$ will implicitly take the security parameter 1^λ as input and omit it stated otherwise.

The following is an illustrative example showing that the equality predicate $\mathsf{Eq}_{\boldsymbol{y}} : \{0,1\}^\ell \to \{0,1\}$ where $\mathsf{Eq}_{\boldsymbol{y}}(\boldsymbol{x}) = 1$ iff $\boldsymbol{y} = \boldsymbol{x}$ can be encoded in a variety of ways into an arithmetic circuit with different properties.

Example. (Encoding Equality Predicates) Let $\mathcal{P}$ be a set of predicates $\{\mathsf{Eq}_{\boldsymbol{y}} | \ \boldsymbol{y} \in \{0,1\}^\ell\}$ where $\mathsf{Eq}_{\boldsymbol{y}}$ is defined as above, and let the input domain be $\mathcal{X} = \{0,1\}^\ell$ where we denote $\mathcal{X} \ni \boldsymbol{x} = (x_1, \cdots, x_\ell)$. We first consider a predicate encoding scheme PES_1 over the finite field $\mathbb{Z}_2$. Namely, for all $\mathsf{Eq}_{\boldsymbol{y}} \in \mathcal{P}$, let $\mathsf{EncPred}(1^\lambda, \mathsf{Eq}_{\boldsymbol{y}})$ output $\hat{C}_{\boldsymbol{y}} : \mathbb{Z}_2^\ell \to \mathbb{Z}_2$ such that $\hat{C}_{\boldsymbol{y}}(\hat{\boldsymbol{x}}) = \Pi_{i\in[\ell]}\big(1 - \hat{x}_i - \hat{y}_i\big)$ where $\hat{\boldsymbol{x}} = \boldsymbol{x} \in \{0,1\}^\ell$ (resp. $\hat{\boldsymbol{y}} = \boldsymbol{y}$) is the output of $\mathsf{EncInpt}(\boldsymbol{x})$ (resp. $\mathsf{EncInpt}(\boldsymbol{y})$). Recalling $-1 = 1$ over $\mathbb{Z}_2$, it can be checked that we have correctness with $S_0 = \{0\}, S_1 = \{1\}$, and the degree of PES_1 is $d = \ell$. Next, we consider a predicate encoding scheme PES_2 over the ring $\mathbb{Z}_q$ with a much lower degree where $d = 1$. In particular, for all $\mathsf{Eq}_{\boldsymbol{y}} \in \mathcal{P}$ and any integer $q > \ell$, let $\mathsf{EncPred}(\mathsf{Eq}_{\boldsymbol{y}})$ output $\hat{C}_{\boldsymbol{y}} : \mathbb{Z}_q^\ell \to \mathbb{Z}_q$ such that $\hat{C}_{\boldsymbol{y}}(\hat{\boldsymbol{x}}) = \ell - \sum_{i\in[\ell]}\big((1 - \hat{y}_i) + (-1 + 2\hat{y}_i) \cdot \hat{x}_i\big)$ where $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \in \{0,1\}^\ell$ are encoded in the same way as above. Now, observing

$$(1 - \hat{y}_i) + (-1 + 2\hat{y}_i) \cdot \hat{x}_i = \hat{x}_i\hat{y}_i + (1 - \hat{x}_i)(1 - \hat{y}_i), \tag{8}$$

it can be checked that $\hat{C}_{\boldsymbol{y}} = 0$ if and only if $\hat{x}_i = \hat{y}_i = 0$ or $\hat{x}_i = \hat{y}_i = 1$, i.e., $\mathsf{Eq}_{\boldsymbol{y}}(\boldsymbol{x}) = 1$. Therefore, we have correctness with $S_0 = \{1, \cdots, \ell\}$, $S_1 = \{0\}$. Furthermore, since $d = 1$, PES_2 is linear. In the following, we continue on to use the left hand form of Eq. (8) to express equality.

Remark 1 (An alternative notion for the input encoding algorithm)**.** *We remark that an alternative more liberal way of defining the* $\mathsf{EncInpt}$ *algorithm is to allow it to encode the input* $\boldsymbol{x}$ *as any element in* $\mathcal{R}$*, rather than only in* $\{0_{\mathcal{R}}, 1_{\mathcal{R}}\}$*. Then, for example, we may create a trivial* PES *scheme with degree* $d = 1$ *for the equality predicate* $\mathsf{Eq}_{\boldsymbol{y}}$ *by encoding elements* $\boldsymbol{x} \in \{0,1\}^{\ell}$ *as an element in* $\mathcal{R} = \mathbb{Z}_{2^\ell}$*, and encoding* $\mathsf{Eq}_{\boldsymbol{y}}$ *as an arithmetic circuit* $\hat{C}_{\boldsymbol{y}}(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}$ *over the ring* $\mathcal{R}$*. However, in this work, we limit ourselves to the stricter notation of simply encoding inputs as 0, 1 elements, since it will provide a more useful and natural compatibility with the algebraic structure of the underlying cryptographic schemes we consider.*

4.2 Encoding Multi-Dimensional Equality Predicates

Here, we propose two predicate encoding schemes for the *multi-dimensional equality predicate*[13] ($\mathsf{MultD\text{-}Eq}$) whose constructions are motivated by different applications. As we show later, the multi-dimensional equality predicate is expressive enough to encode many useful predicates that come up in cryptography (e.g., bit-fixing, subset conjunction, range conjunction predicates), that being for constructions of cryptographic primitives or for embedding secret information during in the security proof.

We first define the domains on which the multi-dimensional equality predicates $\mathsf{MultD\text{-}Eq}$ are defined over, and then formally define what they are.

Definition 8 ($\mathsf{MultD\text{-}Eq}$ Compatible Domains)**.** *Let* p, D, ℓ *be positive integers. We call a pair of domains* $(\mathcal{X}, \mathcal{Y}) \subseteq \mathbb{Z}_p^{D\times\ell} \times \mathbb{Z}_p^{D\times\ell}$ *to be compatible with the multi-dimensional equality predicates if it satisfies the following:*

For all $\mathsf{X} \in \mathcal{X}, \mathsf{Y} \in \mathcal{Y}$ *and for all* $i \in [D]$*, there exists at most one* $j \in [\ell]$ *such that* $\mathsf{X}_{i,j} = \mathsf{Y}_{i,j}$*, where* $\mathsf{X}_{i,j}$ *and* $\mathsf{Y}_{i,j}$ *denotes the* (i,j)*-th element of* X *and* Y *respectively.*

Definition 9 ($\mathsf{MultD\text{-}Eq}$ Predicates)**.** *Let* p, D, ℓ *be positive integers and let* $(\mathcal{X}, \mathcal{Y}) \subseteq \mathbb{Z}_p^{D\times\ell} \times \mathbb{Z}_p^{D\times\ell}$ *be any compatible domains for* $\mathsf{MultD\text{-}Eq}$*. Then, for all* $\mathsf{Y} \in \mathcal{Y}$*, the* multi-dimensional equality predicate $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} : \mathcal{X} \to \{0,1\}$ *is defined as follows:*

$$\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}) = \begin{cases} 1 & \text{if} \quad \forall i \in [D],\ \exists \text{unique } j \in [\ell] \text{ such that } \mathsf{X}_{i,j} = \mathsf{Y}_{i,j} \\ 0 & \text{otherwise} \end{cases},$$

where $\mathsf{X}_{i,j}$ *and* $\mathsf{Y}_{i,j}$ *denotes the* (i,j)*-th element of* X *and* Y *respectively.*

Note that $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X})$ is satisfied only if for each $i \in [D]$, there exists exactly one $j \in [\ell]$ such that $\mathsf{X}_{i,j} = \mathsf{Y}_{i,j}$. Furthermore, since we restrict (X, Y) to be over the compatible domains $(\mathcal{X}, \mathcal{Y})$ for $\mathsf{MultD\text{-}Eq}$, for all $i \in [D]$ we will never have $\mathsf{X}_{i,j} = \mathsf{Y}_{i,j}$ and $\mathsf{X}_{i,j'} = \mathsf{Y}_{i,j'}$ for distinct $j, j' \in [\ell]$. This restriction may appear contrived and inflexible at first, however, this proves to be very useful for constructing predicate encoding schemes with nice qualities, and in fact does not seem to lose much generality in light of expressiveness of the predicate. In particular, by

[13] This predicate is presented in the works of [GMW15] as the $\mathsf{AND\text{-}OR\text{-}EQ}$ predicate satisfying the so called "at most one" promise. We state the conceptual differences between their formalization and ours: they view predicates as functions on both variables X and Y, whereas we view only X as a variable and treat Y as a constant. (Compare [GMW15] Sec. 3.1 and our Def. 9).

appropriately instantiating the compatible domains, we can embed many useful predicates into the $\mathsf{MultD\text{-}Eq}$ predicate. Further discussions are given in the next section and in Appendix B.

We now present two types of predicate encoding schemes for the $\mathsf{MultD\text{-}Eq}$ predicate.

Functionality Preserving Encoding Scheme $\mathsf{PES}_{\mathsf{FP}}$. Our first predicate encoding scheme preserves the functionality of the multi-dimensional equality predicate and can be viewed as an efficient polynomial representation of the circuit computing $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}$.

Lemma 8. *Let $q = q(\lambda), p = p(\lambda), D = D(\lambda), \ell = \ell(\lambda)$ be positive integers and let $(\mathcal{X}, \mathcal{Y}) \subseteq \mathbb{Z}_p^{D\times\ell} \times \mathbb{Z}_p^{D\times\ell}$ be any compatible domains for the $\mathsf{MultD\text{-}Eq}$ predicate. Further, let $\mathcal{P} = \{\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} : \mathcal{X} \to \{0,1\} \mid \mathsf{Y} \in \mathcal{Y}\}$ be a set of $\mathsf{MultD\text{-}Eq}$ predicates. Then the following algorithms $\mathsf{PES}_{\mathsf{FP}} = (\mathsf{EncInpt}_{\mathsf{FP}}, \mathsf{EncPred}_{\mathsf{FP}})$ is a predicate encoding scheme over the ring $\mathbb{Z}_q$ with degree $d = D\zeta$ where $\zeta = \lfloor \log p \rfloor + 1$:*

- $\mathsf{EncInpt}_{\mathsf{FP}}(\mathsf{X}) \to \hat{\mathsf{X}}$: *It takes as input $\mathsf{X} \in \mathcal{X}$, and outputs an encoding $\hat{\mathsf{X}} \in \{0,1\}^{D\ell\zeta}$ as follows:*

$$\hat{\mathsf{X}} = (\mathsf{X}_{i,j,k})_{(i,j,k)\in[D]\times[\ell]\times[\zeta]},$$

  where $\mathsf{X}_{i,j,k}$ is the k-th bit of the binary representation of the (i,j)-th element of X. Here, the output tuple $(\mathsf{X}_{i,j,k})$ is sorted in the lexicographical order. (See Sec. 3.)

- $\mathsf{EncPred}_{\mathsf{FP}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}) \to \hat{C}_{\mathsf{Y}}$: *It takes as input a predicate $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}$, and outputs the following polynomial representation of an arithmetic circuit $\hat{C}_{\mathsf{Y}} : \mathbb{Z}_q^{D\ell\zeta} \to \mathbb{Z}_q$:*

$$\hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}}) = \prod_{i=1}^{D} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - \hat{\mathsf{Y}}_{i,j,k}) + (-1 + 2\hat{\mathsf{Y}}_{i,j,k}) \cdot \hat{\mathsf{X}}_{i,j,k} \Big),$$

  where $\hat{\mathsf{X}}, \hat{\mathsf{Y}} \in \{0,1\}^{D\ell\zeta}$ are encodings of X, Y respectively.

The correctness of $\mathsf{PES}_{\mathsf{FP}}$ holds for the two disjoint subsets $S_0 = \{0\}$, $S_1 = \{1\} \subset \mathbb{Z}_q$.

Proof of Correctness. First, observe that the most inner product equals 1 if $\mathsf{X}_{i,j} = \mathsf{Y}_{i,j}$ and 0 otherwise, due to Eq. (8). Here, recall that $\hat{\mathsf{X}}$ is encoded as $(\mathsf{X}_{i,j,k})$ and $\mathsf{X}_{i,j}$ denotes the (i,j)-th element of X. In the following, denote $\mathsf{X}_i, \mathsf{Y}_i$ as the i-th row of X, Y, respectively. Now, since X and Y come from compatible domains of the $\mathsf{MultD\text{-}Eq}$ predicate, for each $i \in [D]$, we have $\mathsf{X}_{i,j} \neq \mathsf{Y}_{i,j}$ for all $j \in [\ell]$ when $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}_i}(\mathsf{X}_i) = 0$. Therefore, we have the following for all $i \in [D]$:

$$\sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - \hat{\mathsf{Y}}_{i,j,k}) + (-1 + 2\hat{\mathsf{Y}}_{i,j,k}) \cdot \hat{\mathsf{X}}_{i,j,k} \Big) = \begin{cases} 1 & \text{if } \mathsf{MultD\text{-}Eq}_{\mathsf{Y}_i}(\mathsf{X}_i) = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{9}$$

Finally, since $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}) = 1$ if and only if $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}_i}(\mathsf{X}_i) = 1$ for all $i \in [D]$, we have $\hat{C}_{\mathsf{Y}}(\mathsf{X}) = b$ when $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}) = b$ for $b \in \{0,1\}$. Thus, we have $S_0 = \{0\}, S_1 = \{1\}$. □

Linear Encoding Scheme $\mathsf{PES}_{\mathsf{Lin}}$. Our second construction is a linear predicate encoding scheme. It achieves linearity by increasing the length of the encoded input $\hat{\mathsf{X}}$ and takes advantage of the fact that we can change the functionality of the encoded arithmetic circuit $\hat{C}$; the output of $\hat{C}$ can be values other than 0 or 1, whereas outputs of predicates are defined to be in $\{0,1\}$.

Lemma 9. *Let $q = q(\lambda), p = p(\lambda), D = D(\lambda), \ell = \ell(\lambda)$ be positive integers such that $q > D$ and let $(\mathcal{X}, \mathcal{Y}) \subseteq \mathbb{Z}_p^{D\times\ell} \times \mathbb{Z}_p^{D\times\ell}$ be any compatible domains for the* MultD-Eq *predicate. Further, let $\mathcal{P} = \{\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} : \mathcal{X} \to \{0,1\} \mid \mathsf{Y} \in \mathcal{Y}\}$ be a set of* MultD-Eq *predicates. Then the following algorithms $\mathsf{PES}_{\mathsf{Lin}} = (\mathsf{EncInpt}_{\mathsf{Lin}}, \mathsf{EncPred}_{\mathsf{Lin}})$ is a predicate encoding scheme over the ring $\mathbb{Z}_q$ with degree $d = 1$, i.e., a linear scheme, where we set $L = 2^\zeta$ and $\zeta = \lfloor \log p \rfloor + 1$ below.*

- $\mathsf{EncInpt}_{\mathsf{Lin}}(\mathsf{X}) \to \hat{\mathsf{X}}$: *It takes as input $\mathsf{X} \in \mathcal{X}$, and outputs an encoding $\hat{\mathsf{X}} \in \{0,1\}^{D\ell L}$ defined as follows:*

$$\hat{\mathsf{X}} = \Big(\prod_{k=1}^{\zeta} (\mathsf{X}_{i,j,k})^{w_k} \Big)_{(i,j,w)\in[D]\times[\ell]\times[L]},$$

  where w_k and $\mathsf{X}_{i,j,k}$ is the k-th bit of the binary representation of $w - 1$[14] *and the (i,j)-th element of X, respectively. In case $\mathsf{X}_{i,j,k} = w_k = 0$, we define $(\mathsf{X}_{i,j,k})^{w_k}$ to be 1.*

- $\mathsf{EncPred}_{\mathsf{Lin}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}) \to \hat{C}_{\mathsf{Y}}$: *It takes as input a predicate $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}$, and outputs the following polynomial representation of an arithmetic circuit $\hat{C}_{\mathsf{Y}} : \mathbb{Z}_q^{D\ell L} \to \mathbb{Z}_q$:*

$$\hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}}) = D - \sum_{i=1}^{D}\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \hat{\mathsf{X}}_{i,j,w},$$

  where $a_{i,j,w} \in \{-1,0,1\} \subset \mathbb{Z}_q$ is the coefficient for the term $\hat{\mathsf{X}}_{i,j,w} = \prod_{k=1}^{\zeta}(\mathsf{X}_{i,j,k})^{w_k}$ of the polynomial

$$\prod_{k=1}^{\zeta} \Big((1 - \mathsf{Y}_{i,j,k}) + (-1 + 2\mathsf{Y}_{i,j,k}) \cdot \mathsf{X}_{i,j,k} \Big).$$

  Here we treat Y as a constant.

The correctness of $\mathsf{PES}_{\mathsf{Lin}}$ holds for the two disjoint subsets $S_0 = \{1, \cdots, D\}, S_1 = \{0\} \subset \mathbb{Z}_q$.

Proof of Correctness. First, it is easy to check that $a_{i,j,w} \in \{-1,0,1\}$ for all $(i,j,w) \in [D] \times [\ell] \times [L]$, since we have $(1 - \mathsf{Y}_{i,j,k}) \in \{0,1\}$ and $(-1 + 2\mathsf{Y}_{i,j,k}) \in \{-1,1\}$ for any $\mathsf{Y} \in \mathcal{Y}$. The rest of the proof is similar to the previous proof for $\mathsf{PES}_{\mathsf{FP}}$. First, notice that the term $\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \hat{\mathsf{X}}_{i,j,w}$ is the same as the left hand side of Eq. (9). Therefore, we have the following for all $i \in [D]$:

$$\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \hat{\mathsf{X}}_{i,j,w} = \begin{cases} 1 & \text{if } \mathsf{MultD\text{-}Eq}_{\mathsf{Y}_i}(\mathsf{X}_i) = 1 \\ 0 & \text{otherwise} \end{cases},$$

where $\mathsf{X}_i, \mathsf{Y}_i$ are the i-th row of X, Y, respectively. Now, since $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}) = 1$ if and only if $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}_i}(\mathsf{X}_i) = 1$ for all $i \in [D]$, we have the following:

$$\sum_{i=1}^{D}\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \hat{\mathsf{X}}_{i,j,w} = \begin{cases} D & \text{if } \mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}) = 1 \\ \in [0, D-1] & \text{otherwise} \end{cases}.$$

Finally, subtracting the above by D and from the fact that $q > D$, we obtain correctness. □

[14] This inconvenient notion is due to the fact that the bit length of p and L may differ by one in case $p = 2^n - 1$ for $n \in \mathbb{N}$.

Remark 2. *In some applications, the compatible domains* $(\mathcal{X}, \mathcal{Y})$ *for* MultD-Eq *will have some additional structures that we can exploit to obtain more efficient encoding schemes. For an example, in some case for all* $\mathsf{X} \in \mathcal{X}$*, all of the rows of* X *will be equal, i.e.,* $\mathsf{X}_i = \mathsf{X}_{i'}$ *for all* $i, i' \in [D]$ *where* X_i *denotes the* i*-th row of* X*. In this case, we can reduce the output length of* EncInpt *by a factor of* D *by discarding the redundant terms. We will see some concrete examples in the following section and in Appendix B.*

4.3 Expressiveness of Multi-Dimensional Equality Predicates

In this section, we will look at the expressiveness of multi-dimensional equality predicates MultD-Eq. In particular, the following are some predicates that can be expressed as the multi-dimensional equality predicate instantiated with appropriate compatible domains $(\mathcal{X}, \mathcal{Y})$. Combining this with the result of the previous section, we obtain a functionality preserving ($\mathsf{PES}_{\mathsf{FP}}$) or a linear ($\mathsf{PES}_{\mathsf{Lin}}$) encoding scheme for all the following predicates. Note that the choice of the compatible domains are not unique, and different applications would motivate for different constructions. (For further details, see also [BW07, SBC+07, GMW15].) For completeness, in Appendix B.1, we provide discussions on how to obtain the following predicates from the MultD-Eq predicate.

Bit-fixing predicates. For a vector $\mathbf{v} \in \{0, 1, ?\}^\ell$ the bit-fixing predicate $P^{\mathsf{BF}}_{\mathbf{v}} : \{0,1\}^\ell \to \{0,1\}$ is defined as

$$P^{\mathsf{BF}}_{\mathbf{v}}(\mathbf{x}) = 1 \iff \bigwedge_{i=1}^{\ell} \Big((\mathbf{v}_i = \mathbf{x}_i) \bigvee (\mathbf{v}_i = ?) \Big).$$

For example this can be built from MultD-Eq predicates with compatible domains $\mathcal{X}_{\mathsf{BF}}, \mathcal{Y}_{\mathsf{BF}} \subseteq \mathbb{Z}_3^{\ell \times 2}$. This predicate is also known as the *hidden-vector* predicate [BW07].

Equality conjunction predicates. For some finite alphabet Σ and a vector $\mathbf{v} \in \Sigma^\ell$ the equality conjunction predicate $P^{\mathsf{EC}}_{\mathbf{v}} : \Sigma^\ell \to \{0,1\}$ is defined as

$$P^{\mathsf{EC}}_{\mathbf{v}}(\mathbf{x}) = 1 \iff \bigwedge_{i=1}^{\ell} (\mathbf{v}_i = \mathbf{x}_i).$$

For example this can be built from MultD-Eq predicates with compatible domains $\mathcal{X}_{\mathsf{EC}}, \mathcal{Y}_{\mathsf{EC}} \subseteq \mathbb{Z}_{w_1}^{\ell w_2 \times 1}$, where $w_1^{w_2} = |\Sigma|$.

Subset conjunction predicates. For some finite alphabet Σ, let $\mathsf{T}_i \in 2^\Sigma$ for $i \in [\ell]$ and set $\vec{\mathsf{T}} = (\mathsf{T}_1, \cdots, \mathsf{T}_\ell)$. Then the subset conjunction predicate $P^{\mathsf{SC}}_{\vec{\mathsf{T}}} : \prod_{i=1}^{\ell} 2^\Sigma \to \{0,1\}$ is defined as

$$P^{\mathsf{SC}}_{\vec{\mathsf{T}}}(\vec{\mathsf{S}}) = 1 \iff \bigwedge_{i=1}^{\ell} (\mathsf{T}_i \subseteq \mathsf{S}_i),$$

where $\vec{\mathsf{S}} = (\mathsf{S}_1, \cdots, \mathsf{S}_\ell)$. For example this can be built from MultD-Eq predicates with compatible domains $\mathcal{X}_{\mathsf{SC}}, \mathcal{Y}_{\mathsf{SC}} \subseteq \mathbb{Z}_3^{m \times |\Sigma|}$, where $m = \sum_{i=1}^{\ell} |\mathsf{T}_i|$. In particular, when $\ell = 1$, we simply call this predicate as the *subset predicate* $P^{\mathsf{SS}}_{\mathsf{T}} : 2^\Sigma \to \{0,1\}$.

Range conjunction predicates. For $T \in \mathbb{N}$ and $\mathbf{a} = (\mathbf{a}_1, \cdots, \mathbf{a}_\ell), \mathbf{b} = (\mathbf{b}_1, \cdots, \mathbf{b}_\ell) \in [T]^\ell$, the comparison conjunction predicate $P^{\mathsf{RC}}_{[\mathbf{a}:\mathbf{b}]} : [T]^\ell \to \{0,1\}$ is defined as

$$P^{\mathsf{RC}}_{[\mathbf{a}:\mathbf{b}]}(\mathbf{x}) = 1 \iff \bigwedge_{i=1}^{\ell} (\mathbf{a}_i \le \mathbf{x}_i \le \mathbf{b}_i).$$

For example this can be built from MultD-Eq predicates with compatible domains $\mathcal{X}_{\mathsf{RC}}, \mathcal{Y}_{\mathsf{RC}} \subseteq \mathbb{Z}_{T+1}^{\ell \times 2\lceil \log T \rceil}$.

5 Verifiable Random Functions

5.1 Modified Admissible Hash Functions

To construct our VRF, we use the notion of *partitioning function* as introduced in [Yam17], which is a generalization of the standard admissible hash function [BB04b, CHKP10, FHPS13, Jag15]. At a high level, partitioning functions are similar to programmable hash functions, however, unlike programable hash functions that are defined on specific algebraic structures such as bilinear groups [HK08] and lattices [ZCZ16], partitioning functions are purely informational theoretic primitives.

Definition 10 (Partitioning Function)**.** *Let* $\mathsf{F} = \{\mathsf{F}_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}\}_{\lambda \in \mathbb{N}}$ *be a family of functions. We say that* F *is a* partitioning function*, if there exists a PPT algorithm* $\mathsf{PrtSmp}(1^\lambda, Q(\lambda), \epsilon(\lambda))$*, which takes as input a polynomially bounded function* $Q = Q(\lambda)$ *where* $Q : \mathbb{N} \to \mathbb{N}$ *and a noticeable function* $\epsilon = \epsilon(\lambda)$ *where* $\epsilon : \mathbb{N} \to (0, 1/2]$*, and outputs a* partitioning key K *such that*

1. *There exists* $\lambda_0 \in \mathbb{N}$ *such that*

$$\Pr\left[K \in \mathcal{K}_\lambda : K \leftarrow \mathsf{PrtSmp}\Big(1^\lambda, Q(\lambda), \epsilon(\lambda)\Big)\right] = 1$$

   for all $\lambda > \lambda_0$*. Here* λ_0 *may depend on the functions* Q *and* ϵ*.*

2. *For* $\lambda > \lambda_0$*, there exists functions* $\gamma_{\max}(\lambda)$ *and* $\gamma_{\min}(\lambda)$ *that depend on functions* Q *and* ϵ *such that for* $X^{(1)}, \cdots, X^{(Q(\lambda))}, X^* \in \mathcal{X}_\lambda$ *with* $X^* \notin \{X^{(1)}, \cdots, X^{(Q(\lambda))}\}$*,*

$$\gamma_{\min}(\lambda) \leq \Pr\left[\mathsf{F}(K, X^{(1)}) = \cdots = \mathsf{F}(K, X^{(Q(\lambda))}) = 1 \wedge \mathsf{F}(K, X^*) = 0\right] \leq \gamma_{\max}(\lambda) \tag{10}$$

   holds and the function $\tau(\lambda)$ *defined as*

$$\tau(\lambda) := \gamma_{\min}(\lambda) \cdot \epsilon(\lambda) - \frac{\gamma_{\max}(\lambda) - \gamma_{\min}(\lambda)}{2} \tag{11}$$

   is noticeable. The probability is taken over the choice of $K \leftarrow \mathsf{PrtSmp}(1^\lambda, Q(\lambda), \epsilon(\lambda))$*.*

In this work, we consider the particular partitioning function called the *modified admissible hash function*. This allows us to use the same techniques employed by admissible hash functions, while providing for a more compact representation. The following is obtained by the results of [Jag15] and [Yam17].

Definition 11. *(Modified Admissible Hash Function) Let* $n = n(\lambda), \ell = \ell(\lambda)$ *and* $\eta = \eta(\lambda)$ *be an integer-valued function of* λ *such that* $n, \ell = \Theta(\lambda)$ *and* $\eta = \omega(\log \lambda)$*, and* $\{C_n : \{0,1\}^n \to \{0,1\}^\ell\}_{n \in \mathbb{N}}$ *be a family of error correcting codes with minimal distance* $c \cdot \ell$ *for a constant* $c \in (0, 1/2)$*. Let*

$$\mathcal{K}_{\mathsf{MAH}} = \{\mathsf{T} \subseteq [2\ell] \mid |\mathsf{T}| < \eta\} \quad \textit{and} \quad \mathcal{X}_{\mathsf{MAH}} = \{0,1\}^n.$$

Then, we define the modified admissible hash function $\mathsf{F}_{\mathsf{MAH}} : \mathcal{K}_{\mathsf{MAH}} \times \mathcal{X}_{\mathsf{MAH}} \to \{0,1\}$ *as*

$$\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = \begin{cases} 0, & \text{if } \mathsf{T} \subseteq \mathsf{S}(X) \\ 1, & \text{otherwise} \end{cases} \quad \text{where} \quad \mathsf{S}(X) = \{2i - C(X)_i \mid i \in [\ell]\}. \tag{12}$$

In the above, $C(X)_i$ *is the* i*-th bit of* $C(X) \in \{0,1\}^\ell$.

Theorem 1. *There exists an efficient algorithm* $\mathsf{PrtSmp}_{\mathsf{MAH}}(1^\lambda, Q(\lambda), \epsilon(\lambda))$ *which takes as input a polynomially bounded function* $Q = Q(\lambda)$ *where* $Q : \mathbb{N} \to \mathbb{N}$ *and a noticeable function* $\epsilon = \epsilon(\lambda)$ *where* $\epsilon : \mathbb{N} \to (0, 1/2]$*, and outputs* T *with cardinality exactly* $\eta' = \eta'(\lambda)$*, where*

$$\eta' := \left\lfloor \frac{\log(2Q + Q/\epsilon)}{-\log(1-c)} \right\rfloor,$$

such that Eq. (10) *and* (11) *hold with respect to* $\mathsf{F} := \mathsf{F}_{\mathsf{MAH}}, \mathsf{PrtSmp} := \mathsf{PrtSmp}_{\mathsf{MAH}}$ *and* $\tau(\lambda) = 2^{-\eta'-1} \cdot \epsilon$*. In particular,* $\mathsf{F}_{\mathsf{MAH}}$ *is a partitioning function.*

5.2 Construction

Intuition. In our VRF construction, we implicitly embed the partitioning function $\mathsf{F}_{\mathsf{MAH}}$ in the output Y during simulation. In particular, as we mentioned in the technical overview, the strategy is to embed $\mathsf{F}_{\mathsf{MAH}}$ in the exponent of g using an L-DDH assumption with the smallest possible L.

Since $\mathsf{F}_{\mathsf{MAH}}$ checks whether the subset predicate $P^{\mathsf{Sub}}_{\mathsf{T}} : 2^{[2\ell]} \to \{0,1\}$ is satisfied or not, which is a special case of the subset conjunction predicate presented in Sec. 4.3, it can be encoded as the multi-dimensional equality predicate $\mathsf{MultD\text{-}Eq}$ with (exploitable) compatible domains $\mathcal{X}_{\mathsf{Sub}}, \mathcal{Y}_{\mathsf{Sub}} \subseteq \mathbb{Z}_{2\ell}^{|\mathsf{T}| \times \ell}$ as we show in Appendix B. For our VRF construction, we consider the functionality preserving encoding scheme $\mathsf{PES}_{\mathsf{FP}}$ for this $\mathsf{MultD\text{-}Eq}$ predicate with compatible domains $(\mathcal{X}_{\mathsf{Sub}}, \mathcal{Y}_{\mathsf{Sub}})$, and set up the verification keys so that the following polynomial is implicitly embedded during the security reduction:

$$\hat{C}^{\mathsf{Sub}}_{\mathsf{T}}(\hat{\mathsf{S}}) = \prod_{i=1}^{\eta} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - \hat{\mathsf{S}}_{j,k}) + (-1 + 2\hat{\mathsf{S}}_{j,k}) \cdot \hat{\mathsf{T}}_{i,k} \Big), \tag{13}$$

where (informaly) $\hat{\mathsf{T}}_{i,k}, \hat{\mathsf{S}}_{j,k}$ corresponds to the k-th bit of the binary representation of the i-th and j-th element in the set $\mathsf{T}, \mathsf{S} \subseteq [2\ell]$, respectively. From the correctness of $\mathsf{PES}_{\mathsf{FP}}$, we have $\hat{C}^{\mathsf{Sub}}_{\mathsf{T}}(\hat{\mathsf{S}}) = (\mathsf{T} \subseteq \mathsf{S})$ as desired. Here, recall that the set $\mathsf{S} = \mathsf{S}(X)$ is uniquely constructed for each input $X \in \{0,1\}^n$. Finally, since $\eta = \omega(\log \lambda)$, $\ell = \Theta(\lambda), \zeta = \Theta(\log \lambda)$, the above polynomial will be of degree $\omega(\log^2 \lambda)$. Thus, this allows us to simulate the proof π and evaluation Y (and hence prove security of our VRF) under a L-DDH assumption where $L = \omega(\log^2 \lambda)$.

Construction. For simplicity of presentation, we deviate slightly from the notations used above. Below, $n, \ell, \eta, \mathsf{S}(\cdot)$ are the parameters and function specified by the modified admissible hash function (Def. 11) and ζ is set as $\lfloor \log p \rfloor + 1$.

$\mathsf{Gen}(1^\lambda)$: On input 1^λ, it runs $\Pi \leftarrow \mathsf{GrpGen}(1^\lambda)$ to obtain a group description. It then chooses random generators $g, h \leftarrow \mathbb{G}^*$ and $w_0, w_{i,k} \leftarrow \mathbb{Z}_p$ for $(i,k) \in [\eta] \times [\zeta]$. Finally, it outputs

$$\mathsf{vk} = \left(\Pi, g, h, g_0 := g^{w_0}, \left(g_{i,k} := g^{w_{i,k}} \right)_{(i,k) \in [\eta] \times [\zeta]} \right), \quad \text{and} \quad \mathsf{sk} = \left(w_0, (w_{i,k})_{(i,k) \in [\eta] \times [\zeta]} \right).$$

$\mathsf{Eval}(\mathsf{sk}, X)$: On input $X \in \{0,1\}^n$, it first computes $\mathsf{S}(X) = \{s_1, \cdots, s_\ell\} \in [2\ell]$. In the following, let $s_{j,k}$ be the k-th bit of the binary representation of s_j, where $k \in [\zeta]$. It then computes

$$\theta_{i'} = \prod_{i=1}^{i'} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot w_{i,k} \Big), \quad \text{and} \quad \theta_{i,j,k'} = \prod_{k=1}^{k'} \Big((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot w_{i,k} \Big)$$

for $i' \in [\eta]$ and $(i, j, k') \in [\eta] \times [\ell] \times [\zeta]$, and defines $\theta := \theta_\eta$. Finally, it outputs

$$Y = e(g,h)^{\theta/w_0}, \quad \text{and} \quad \pi = \left(\pi_0 := g^{\theta/w_0}, \left(\pi_{i'} := g^{\theta_{i'}} \right)_{i' \in [\eta]}, \left(\pi_{i,j,k'} := g^{\theta_{i,j,k'}} \right)_{(i,j,k') \in [\eta] \times [\ell] \times [\zeta]} \right).$$

$\mathsf{Verify}(\mathsf{vk}, X, (Y, \pi))$: First, it checks the validity of vk. It outputs 0 if any of the following properties are not satisfied.

1. vk is of the form $\left(\Pi, g, h, g_0, (g_{i,k})_{(i,k) \in [\eta] \times [\zeta]} \right)$.
2. $\mathsf{GrpVfy}(\Pi) = 1$ and $\mathsf{GrpVfy}(\Pi, s) = 1$ for all $s \in (g, h, g_0) \cup (g_{i,k})_{(i,k) \in [\eta] \times [\zeta]}$.

Then, it checks the validity of X, Y and π. In doing so, it first prepares the terms $\Phi_{i'}, \bar{g}_{i,j,k'}$ for all $i' \in [\eta], (i, j, k') \in [\eta] \times [\ell] \times [\zeta]$ defined as

$$\Phi_{i'} := \prod_{j=1}^{\ell} \pi_{i',j,\zeta}, \quad \text{and} \quad \bar{g}_{i,j,k'} := g^{1 - s_{j,k'}} \cdot (g_{i,k'})^{-1 + 2s_{j,k'}}.$$

It outputs 0 if any of the following properties are not satisfied.

3. $X \in \{0,1\}^n, Y \in \mathbb{G}_T$, π is of the form $\left(\pi_0, (\pi_{i'})_{i' \in [\eta]}, (\pi_{i,j,k'})_{(i,j,k') \in [\eta] \times [\ell] \times [\zeta]} \right)$.
4. It holds that for all $i' \in [\eta - 1]$ and $(i, j, k') \in [\eta] \times [\ell] \times [\zeta - 1]$,

$$\begin{aligned} e(\pi_1, g) &= e(\Phi_1, g), & e(\pi_{i,j,1}, g) &= e(\bar{g}_{i,j,1}, g), \\ e(\pi_{i'+1}, g) &= e(\Phi_{i'+1}, \pi_{i'}), & e(\pi_{i,j,k'+1}, g) &= e(\bar{g}_{i,j,k'+1}, \pi_{i,j,k'}). \end{aligned}$$

5. It holds that $e(\pi_\eta, g) = e(\pi_0, g_0)$ and $e(\pi_0, h) = Y$.

If all the above checks are passed, it outputs 1.

5.3 Correctness, Unique Provability, and Pseudorandomness

Theorem 2 (Correctness and Unique Provability)**.** *Our scheme forms a correct verifiable random function and satisfies the unique provability requirement.*

Proof. We first prove the correctness of the scheme. It is easily seen that when Gen and Eval are properly run, then it passes Step 1, 2, 3 of the verification algorithm. Next, observe that

$$\begin{aligned} \Phi_{i'} = \prod_{j=1}^{\ell} \pi_{i',j,\zeta} = g^{\sum_{j=1}^{\ell} \theta_{i',j,\zeta}} = g^{\sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \left((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot w_{i',k} \right)} \\ \bar{g}_{i,j,k'} = g^{1 - s_{j,k'}} \cdot (g_{i,k'})^{-1 + 2s_{j,k'}} = g^{\left((1 - s_{j,k'}) + (-1 + 2s_{j,k'}) \cdot w_{i,k'} \right)}, \end{aligned}$$

for all $i' \in [\eta]$ and $(i,j,k') \in [\eta] \times [\ell] \times [\zeta]$. Since $\Phi_1 = \pi_1$ and $\bar{g}_{i,j,1} = \pi_{i,j,1}$, the first two equation in Step 4 holds. The equality of the rest of the equations in Step 4 follow using the additional observation that

$$\begin{cases} \theta_{i'+1} = \theta_{i'} \cdot \Big(\sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \big((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot w_{i'+1,k}\big) \Big) \\ \theta_{i,j,k'+1} = \theta_{i,j,k'} \cdot \big((1 - s_{j,k'+1}) + (-1 + 2s_{j,k'+1}) \cdot w_{i,k'+1}\big) \end{cases},$$

for all $i' \in [\eta - 1]$ and $(i,j,k') \in [\eta] \times [\ell] \times [\zeta - 1]$. Finally, since by definition $\pi_\eta = g^{\theta_\eta} = g^{\theta}$, Step 5 holds. This completes the proof of the correctness of the scheme.

Next, we turn to prove the unique provability of the scheme. We have to show that for any $\mathsf{vk} \in \{0,1\}^*$ and $X \in \{0,1\}^n$, there does not exist any (Y_0, π_0, Y_1, π_1) such that $Y_0 \neq Y_1$ and $\mathsf{Verify}(\mathsf{vk}, X, (Y_0, \pi_0)) = \mathsf{Verify}(\mathsf{vk}, X, (Y_1, \pi_1)) = 1$.

- First of all, in Step 1 and Step 2, the verification algorithm checks whether Π contains valid certified bilinear group parameters and checks whether all group elements $g, h, g_0, (g_{i,j})_{(i,j)\in[\eta]\times[\zeta]}$ are valid group elements with respect to Π. Thus, in the following, we may assume that all these group elements are valid and have a unique encoding.

- In Step 3, it is checked whether $X \in \{0,1\}^n$, $Y \in \mathbb{G}_T$ and π is in the proper form. In Step 4 and Step 5 it inductively checks whether all the equality holds. Now, since the bilinear group is satisfied, i.e., each group element has a unique encoding and the bilinear map is non-degenerate, there exists only one unique π such that correctness holds.

Therefore, the value of (Y, π) is uniquely determined by the input X and the verification key vk. This completes the proof. □

Theorem 3 (Pseudorandomness)**.** *Our scheme satisfies pseudorandomness assuming* L-DDH *with* $L = \eta\zeta = \omega(\log^2 \lambda)$.

Proof. Let $\mathcal{A}$ be a PPT adversary that breaks the pseudorandomness of the scheme with non-negligible advantage. Let $\epsilon = \epsilon(\lambda)$ be its advantage and $Q = Q(\lambda)$ be the upper bound on the number of evaluation queries it makes. Here, since $\mathcal{A}$ is a valid adversary, Q is a polynomially bounded function and there exists a noticeable function $\epsilon_0 = \epsilon_0(\lambda)$ such that $\epsilon(\lambda) \geq \epsilon_0(\lambda)$ holds for infinitely many λ. Then combining Def. 10 and Thm. 1 together, for $\mathsf{T} \leftarrow \mathsf{PrtSmp}_{\mathsf{MAH}}(1^\lambda, Q(\lambda), \epsilon_0(\lambda))$, we have $\mathsf{T} \subseteq [2\ell]$ and $|\mathsf{T}| < \eta$ with probability 1 for all sufficiently large λ. Therefore, in the following, we assume this condition always holds. We show security of the scheme through a sequence of games. In each game, a value $\mathsf{coin}' \in \{0,1\}$ is defined. While it is set $\mathsf{coin}' = \widehat{\mathsf{coin}}$ in the first game, these values might be different in the later games. In the following we define E_i to be the event that $\mathsf{coin}' = \mathsf{coin}$ in Game_i.

Game_0 : This is the actual security game. Since $\mathcal{Y} = \mathbb{G}_T$, when $\mathsf{coin} = 1$, a random element $Y_1^* \leftarrow \mathbb{G}_T$ is returned to $\mathcal{A}$ as the challenge query. At the end of the game, $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}}$ for coin. Finally, the challenger sets $\mathsf{coin}' = \widehat{\mathsf{coin}}$. By assumption on the adversary $\mathcal{A}$, we have

$$\left|\Pr[\mathsf{E}_0] - \frac{1}{2}\right| = \left|\Pr[\mathsf{coin}' = \mathsf{coin}] - \frac{1}{2}\right| = \left|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}\right| = \epsilon.$$

Game_1 : In this game, we change Game_0 so that the challenger performs an additional step at the end of the game. Namely, the challenger first runs $\mathsf{T} \leftarrow \mathsf{PrtSmp}_{\mathsf{MAH}}(1^\lambda, Q(\lambda), \epsilon_0(\lambda))$ from Thm. 1. As noted earlier, we have $|\mathsf{T}| \subseteq [2\ell]$ and $|\mathsf{T}| < \eta$. Then, it checks whether the following condition holds:

$$\begin{aligned} &\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^{(1)}) = 1 \ \wedge \ \cdots = \ \wedge \ \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^{(Q)}) = 1 \ \wedge \ \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^*) = 0 \\ \iff \ &\Big(\mathsf{T} \not\subseteq \mathsf{S}(X^{(1)})\Big) \ \wedge \ \cdots \ \wedge \ \Big(\mathsf{T} \not\subseteq \mathsf{S}(X^{(Q)})\Big) \ \wedge \ \Big(\mathsf{T} \subseteq \mathsf{S}(X^*)\Big) \end{aligned} \tag{14}$$

where X^* is the challenge input and $\{X^{(i)}\}_{i\in[Q]}$ are the inputs for which $\mathcal{A}$ has queried the evaluation of the function. If it does not hold, the challenger ignores the output $\widehat{\mathsf{coin}}$ of $\mathcal{A}$ and sets $\mathsf{coin}' \leftarrow \{0,1\}$. In this case, we say that the challenger aborts. If condition (14) holds, the challenger sets $\mathsf{coin}' = \widehat{\mathsf{coin}}$. By Lem. 7 and Thm. 1 (See also Def. 10, Item 2), the following holds for infinitely many λ:

$$\begin{aligned} \left|\Pr[\mathsf{E}_1] - \frac{1}{2}\right| &\geq \gamma_{\min} \cdot \epsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2} \\ &\geq \gamma_{\min} \cdot \epsilon_0 - \frac{\gamma_{\max} - \gamma_{\min}}{2} \\ &\geq \tau, \end{aligned}$$

where $\tau = \tau(\lambda)$ is a noticeable function. Recall that $\gamma_{\max}, \gamma_{\min}, \tau$ are functions specified by Q, ϵ_0 and the underlying partitioning function $\mathsf{F}_{\mathsf{MAH}}$.

Game_2 : In this game, we change the way $w_0, (w_{i,k})_{(i,k)\in[\eta]\times[\zeta]}$ are chosen. First, at the beginning of the game, the challenger picks $\mathsf{T} \leftarrow \mathsf{PrtSmp}_{\mathsf{MAH}}(1^\lambda, Q(\lambda), \epsilon_0(\lambda))$ and parses it as $\mathsf{T} = \{t_1, \cdots, t_{\eta'}\} \subset [2\ell]$. Note that changing the time on which the adversary runs the algorithm is only conceptual. Now, recalling that by our assumption $\eta' < \eta$, it sets $t_i = 0$ for $i \in [\eta' + 1, \eta]$. Next, it samples $\alpha \leftarrow \mathbb{Z}_p^*$ and $\tilde{w}_0, \tilde{w}_{i,k} \leftarrow \mathbb{Z}_p$ for $(i,k) \in [\eta] \times [\zeta]$. Finally, the challenger sets

$$w_0 = \tilde{w}_0 \cdot \alpha, \quad w_{i,k} = \tilde{w}_{i,k} \cdot \alpha + t_{i,k} \quad \text{for} \quad (i,k) \in [\eta] \times [\zeta], \tag{15}$$

where $t_{i,k}$ is the k-th bit of the binary representation of t_i. The rest of the game is identical to Game_1. Here, the statistical distance of the distributions of $w_0, (w_{i,k})_{(i,k)\in[\eta]\times[\zeta]}$ in Game_1 and Game_2 is at most $(\eta\zeta + 1)/p$, which is negligible. Therefore, we have

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda).$$

Before, getting into Game_3, we introduce polynomials (associated with each input X) that implicitly embeds the information on the partitioning function $\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X)$, i.e., the form of the polynomials depend on whether $\mathsf{T} \subseteq \mathsf{S}(X)$ or not. For any $\mathsf{T} \subseteq [2\ell]$ with $|\mathsf{T}| = \eta' < \eta$ and $X \in \{0,1\}^n$ (i.e., for any $\mathsf{S}(X)$), we define the polynomial $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) : \mathbb{Z}_p \to \mathbb{Z}_p$ as

$$\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) = \prod_{i=1}^{\eta} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot (\tilde{w}_{i,k}\mathsf{Z} + t_{i,k})\Big), \tag{16}$$

where $\{s_{j,k}\}_{(j,k)\in[\ell]\times[\zeta]}$ and $\{t_{i,k}\}_{(i,k)\in[\eta]\times[\zeta]}$ are defined as in Game_2. Note that $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\alpha) = \theta$. Our security proof is built upon the following lemma on the partitioning function.

Lemma 10. *There exists* $\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) : \mathbb{Z}_p \to \mathbb{Z}_p$ *such that*

$$\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) = \begin{cases} 1 + \mathsf{Z}\cdot\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}), & \text{if } \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = 0 \\ \mathsf{Z}\cdot\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}), & \text{if } \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = 1 \end{cases}.$$

In other words, $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z})$ *is not divisible by* Z *if and only if* $\mathsf{T} \subseteq \mathsf{S}(X)$.

So as not to interrupt the proof of Thm. 3, we intentionally skip the proof of Lem. 10 for the time being. Furthermore, with an abuse of notation, for all $X \in \{0,1\}^n$, we define the following polynomials that map $\mathbb{Z}_p$ to $\mathbb{Z}_p$, which are defined analogously to the values computed during Eval:

$$\begin{cases} \theta^X_{i'}(\mathsf{Z}) = \prod_{i=1}^{i'} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1-s_{j,k}) + (-1+2s_{j,k})(\tilde{w}_{i,k}\mathsf{Z} + t_{i,k})\Big) \\ \theta^X_{i,j,k'}(\mathsf{Z}) = \prod_{k=1}^{k'} \Big((1-s_{j,k}) + (-1+2s_{j,k})(\tilde{w}_{i,k}\mathsf{Z} + t_{i,k})\Big) \end{cases},$$

for $i' \in [\eta]$ and $(i,j,k') \in [\eta]\times[\ell]\times[\zeta]$, and define $\theta^X(\mathsf{Z}) := \theta^X_\eta(\mathsf{Z})$. Note that we have $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) = \theta^X(\mathsf{Z}), \theta_{i'} = \theta^X_{i'}(\alpha), \theta_{i,j,k'} = \theta^X_{i,j,k'}(\alpha)$, and $\theta = \theta^X(\alpha)$.

Game_3 : Recall that in the previous game, the challenger aborts at the end of the game if condition (14) is not satisfied. In this game, we change the game so that the challenger aborts as soon as the abort condition becomes true. Since this is only a conceptual change, we have

$$\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3].$$

Game_4 : In this game, we change the way the evaluation queries are answered. When the adversary $\mathcal{A}$ queries an input X to be evaluated, it first checks whether $\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = 1$, i.e., it checks if condition (14) is satisfied. If it does not hold, it aborts as in Game_3. Otherwise, it computes the polynomial $\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ such that $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) = \mathsf{Z}\cdot\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z})$, and returns

$$Y = e(g^{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\alpha)/\tilde{w}_0}, h), \tag{17}$$

$$\pi = \left(\pi_0 = g^{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\alpha)/\tilde{w}_0}, \left(\pi_{i'} = g^{\theta^X_{i'}(\alpha)}\right)_{i'\in[\eta]}, \left(\pi_{i,j,k'} = g^{\theta^X_{i,j,k'}(\alpha)}\right)_{(i,j,k')\in[\eta]\times[\ell]\times[\zeta]}\right). \tag{18}$$

Note that existence of such a polynomial $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z})$ is guaranteed by Lem. 10. By the definition of $\theta^X_{i'}(\mathsf{Z})$ and $\theta^X_{i,j,k'}(\mathsf{Z})$, the components $\pi_{i'}$ and $\pi_{i,j,k'}$ are correctly generated. Furthermore, we have

$$\frac{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\alpha)}{\tilde{w}_0} = \frac{\alpha\cdot\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\alpha)}{\alpha\cdot\tilde{w}_0} = \frac{\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\alpha)}{w_0} = \frac{\theta}{w_0}.$$

Therefore, Y and π_0 are also correctly generated, and the challenger simulates the evaluation queries perfectly. Hence,

$$\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4].$$

Game_5 : In this game, we change the way the challenge ciphertext is created when $\mathsf{coin} = 0$. Recall in the previous games when $\mathsf{coin} = 0$, we created a valid $Y_0^* = \mathsf{Eval}(\mathsf{sk}, X^*)$ as in the real scheme. If $\mathsf{coin} = 0$ and $\mathsf{F}_{\mathsf{MAH}}(X^*) = 0$ (i.e., if it does not abort), to create Y_0^*, the challenger first computes the polynomial $\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{X}]$ such that $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\mathsf{Z}) = 1 + \mathsf{Z}\cdot\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\mathsf{Z})$, whose existence is guaranteed by Lem. 10. It then sets,

$$Y_0^* = \left(e(g,h)^{1/\alpha}\cdot e(g,h)^{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha)}\right)^{1/\tilde{w}_0}$$

and returns it to $\mathcal{A}$. Here, the above term can be written equivalently as

$$\left(e(g,h)^{1/\alpha}\cdot e(g,h)^{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha)}\right)^{1/\tilde{w}_0} = e(g^{(1+\alpha\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha))/\alpha\tilde{w}_0}, h) = e(g^{\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha)/w_0}, h) = e(g^{\theta/w_0}, h).$$

Therefore, the view of the adversary in unchanged. Hence,

$$\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5].$$

Game_6 : In this game, we change the challenge value to be a random value in $\mathbb{G}_T$ regardless of whether $\mathsf{coin} = 0$ of $\mathsf{coin} = 1$. Namely, the challenger sets $Y^* \leftarrow \mathbb{G}_T$. As we will show in Lem. 11, assuming L-DDH is hard for $L = \eta\zeta$, we have $|\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6]| = \mathsf{negl}(\lambda)$.

Analysis. From the above, we have

$$\begin{aligned}\left|\Pr[\mathsf{E}_6] - \frac{1}{2}\right| &= \left|\Pr[\mathsf{E}_1] - \frac{1}{2} + \sum_{i=1}^{5}(\Pr[\mathsf{E}_{i+1}] - \Pr[\mathsf{E}_i])\right| \\ &\geq \left|\Pr[\mathsf{E}_1] - \frac{1}{2}\right| - \sum_{i=1}^{5}|\Pr[\mathsf{E}_{i+1}] - \Pr[\mathsf{E}_i]| \\ &\geq \tau(\lambda) - \mathsf{negl}(\lambda), \end{aligned} \tag{19}$$

for infinitely many λ. Since $\Pr[\mathsf{E}_6] = 1/2$, this implies $\tau(\lambda) \leq \mathsf{negl}(\lambda)$ for infinitely many λ, which is a contradiction. □

To complete the proof of Thm. 3, it remains to prove Lem. 10 and 11.

Proof of Lem. 10. First, we can rewrite Eq.(16) as

$$\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(\mathsf{X})}(\mathsf{Z}) = \mathsf{Z}\cdot\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(\mathsf{X})}(\mathsf{Z}) + \underbrace{\prod_{i=1}^{\eta}\sum_{j=1}^{\ell}\prod_{k=1}^{\zeta}\Big((1-s_{j,k}) + (-1+2s_{j,k})\cdot t_{i,k}\Big)}_{=C},$$

for some polynomial $\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(\mathsf{X})}(\mathsf{Z})$ with degree at most $\eta\zeta$. Observe the constant term C is computing the circuit outputted by the functionality preserving encoding scheme $\mathsf{PES}_{\mathsf{FP}}$ for the subset predicate (See Lem. 8 and Appendix B.2). Therefore, we have $C = 1$ if $\mathsf{T} \subseteq \mathsf{S} \Leftrightarrow \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = 0$, and $C = 0$ otherwise, as desired. □

Lemma 11. *For any PPT adversary $\mathcal{A}$, there exists another PPT adversary $\mathcal{B}$ such that*

$$|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]| \leq \mathsf{Adv}_{\mathcal{B}}^{L\text{-}DDH},$$

where $L = \eta\zeta$. In particular, under the L-DDH assumption, we have $|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]| = \mathsf{negl}(n)$.

Proof. Suppose an adversary $\mathcal{A}$ that has non-negligible advantage in distinguishing Game_5 and Game_6. We use $\mathcal{A}$ to construct an *L-DDH* algorithm denoted $\mathcal{B}$, which proceeds as follows.

Instance. $\mathcal{B}$ is given the problem instance of $L\text{-}DDH(\Pi, g, h, \{g^{\alpha^i}\}_{i\in L}, \Psi)$ for $L = \eta\zeta$, where $\Psi = e(g,h)^{1/\alpha}$ or $\Psi \leftarrow \mathbb{G}_T$.

Setup. To construct the verification key vk, it samples $\tilde{w}_0, \tilde{w}_{i,k} \leftarrow \mathbb{Z}_p$ for $(i,k) \in [\eta] \times [\zeta]$ as in Game_2, and implicitly sets w_0 and $(w_{i,k})_{(i,k)\in[\eta]\times[\zeta]}$ as in Eq. (15). Then, since $w_0, (w_{i,k})_{(i,k)\in[\eta]\times[\zeta]}$ are all at most degree 1 polynomials in α, $\mathcal{B}$ can efficiently compute $g_0, (g_{i,k})_{(i,k)\in[\eta]\times[\zeta]}$ from the problem instance. Finally, it returns $\mathsf{vk} = (\Pi, g, h, g_0, (g_{i,k})_{(i,k)\in[\eta]\times[\zeta]})$ to $\mathcal{A}$. $\mathcal{B}$ also picks a random bit $\mathsf{coin} \leftarrow \{0,1\}$ and keeps it secret.

Phase 1 and Phase 2. The evaluation queries made by $\mathcal{A}$ are answered as in Eq. (17) of Game_4. Observe that this can be done efficiently, since $\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z})$ is degree at most $L = \eta\zeta$.

Challenge Query. When $\mathcal{A}$ makes the challenge query for the challenge input X^*, $\mathcal{B}$ first computes $\mathsf{F}_{\mathsf{MAH}}(X^*)$. Then, it aborts and sets $\mathsf{coin}' \leftarrow \{0,1\}$ if $\mathsf{F}_{\mathsf{MAH}}(X^*) = 1$. Otherwise, it proceeds as follows. If $\mathsf{coin} = 0$, it computes

$$Y_0^* = \left(\Psi \cdot e(g,h)^{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha)}\right)^{1/\tilde{w}_0}.$$

Note that $e(g,h)^{\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha)}$ can be efficiently computed from the problem instance, since $\mathsf{R}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\mathsf{Z})$ is of degree at most $L = \eta\zeta$. In the case of $\mathsf{coin} = 1$, $\mathcal{B}$ sets $Y^* \leftarrow \mathbb{G}_T$. In both cases, $\mathcal{B}$ returns Y^* to $\mathcal{A}$.

Guess. At last, $\mathcal{A}$ outputs its guess $\widehat{\mathsf{coin}}$ (if the abort condition has not been satisfied). Then, $\mathcal{B}$ sets $\mathsf{coin}' = \widehat{\mathsf{coin}}$. Finally, $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$ and 0 otherwise.

Analysis. It can be seen that $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in Game_5 if $\Psi = e(g,h)^{1/\alpha}$ and Game_6 if $\Psi \leftarrow \mathbb{G}_T$. We therefore conclude that

$$\mathsf{Adv}_{\mathcal{B}}^{L\text{-}DDH} = |\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]|$$

for $L = \eta\zeta$ as desired.

□

5.4 Achieving Smaller Proof Size

In this section, we propose a variant of the VRF presented in Sec. 5.2 with a much shorter proof size. Recall the VRF we constructed in the previous section had a very small verification key size $|\mathsf{vk}| \approx \eta\zeta = \omega(\log^2\lambda)$ with a rather large proof size $|\pi| \approx \eta\ell\zeta = \omega(\lambda\log^2\lambda)$, where we count the number of group elements for size. A first attempt is to use a similar trick used in [Yam17] to add some helper terms in the verification key to make the proof size smaller. In particular, we can convert our VRF to have a very small proof size $|\pi| = \omega(\log\lambda)$ by allowing the verification key size to grow quasi-linearly in the security parameter, i.e., $|\mathsf{vk}| = \omega(\lambda\log\lambda)$.

However, we can do much better with an additional idea; we obtain a VRF with proof size $|\pi| = \omega(\log\lambda)$ and verification key size $|\mathsf{vk}| = \omega(\sqrt{\lambda}\log\lambda)$, which is now sublinear.

Preparation. We define *power tuples* $\mathcal{P}(W)$ for a tuple W, analogously to power sets. Namely, we create a tuple that contains all the subsequence of W in lexicographical order, i.e., $\mathcal{P}(W) = (w_1, w_2, w_3, w_1w_2, w_1w_3, w_2w_3, w_1w_2w_3)$ for $W = (w_1, w_2, w_3)$. Here, we do not consider the empty string as a subsequence of W. For a group element $g \in \mathbb{G}$ or $\mathbb{G}_T$ and a tuple W with elements in $\mathbb{Z}_p$, we denote $g^{\mathcal{P}(W)}$ as the tuple $(g^w \mid w \in \mathcal{P}(W))$. Furthermore, for tuples W, W' with elements

in $\mathbb{Z}_p$ we define $e(g^{\mathcal{P}(W)}, g^{\mathcal{P}(W')})$ to be the tuple $(e(g,g)^{ww'} \mid w \in W, w' \in W')$. Assume all the tuples are sorted in the lexicographical order.

Construction. Below, we provide a VRF with small proof size.

$\mathsf{Gen}(1^\lambda)$: On input 1^λ, it runs $\Pi \leftarrow \mathsf{GrpGen}(1^\lambda)$ to obtain a group description. It then chooses random generators $g, h \leftarrow \mathbb{G}^*$, $w_0, w_{i,k} \leftarrow \mathbb{Z}_p$ for $(i,k) \in [\eta] \times [\zeta]$ and sets $L_i = (w_{i,k})_{k \in [\lfloor \zeta/2 \rfloor]}$ and $R_i = (w_{i,k})_{k \in [\lfloor \zeta/2 \rfloor + 1, \zeta]}$. Finally, it outputs

$$\mathsf{vk} = \left(\Pi, g, h, g_0 := g^{w_0}, (g^{\mathcal{P}(L_i)}, g^{\mathcal{P}(R_i)})_{i \in [\eta]}\right), \quad \text{and} \quad \mathsf{sk} = \left(w_0, (w_{i,k})_{(i,k) \in [\eta] \times [\zeta]}\right).$$

Note that we have $e(g^{\mathcal{P}(L_i)}, g^{\mathcal{P}(R_i)}) = e(g,g)^{\mathcal{P}(W_i)}$ where $W_i = (w_{i,k})_{k \in [\zeta]}$.

$\mathsf{Eval}(\mathsf{sk}, X)$: On input $X \in \{0,1\}^n$, it first computes $\mathsf{S}(X) = \{s_1, \cdots, s_\ell\} \in [2\ell]$. In the following, let $s_{j,k}$ be the k-th bit of the binary representation of s_j, where $k \in [\zeta]$. It then computes

$$\begin{cases} \theta_i = \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \left((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot w_{i,k}\right) \\ \theta_{[1:i']} = \prod_{i=1}^{i'} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \left((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot w_{i,k}\right) \end{cases},$$

for $i \in [\eta], i' \in [2, \eta]$ and sets $\theta := \theta_{[1:\eta]}$. Note that we do not require $i' = 1$ since $\theta_1 = \theta_{[1:1]}$. Finally, it outputs

$$Y = e(g,h)^{\theta/w_0}, \quad \text{and} \quad \pi = \left(\pi_0 := g^{\theta/w_0}, \left(\pi_i := g^{\theta_i}\right)_{i \in [\eta]}, \left(\pi_{[1:i']} := g^{\theta_{[1:i']}}\right)_{i' \in [2,\eta]}\right).$$

$\mathsf{Verify}(\mathsf{vk}, X, (Y, \pi))$: First, it checks the validity of vk. It outputs 0 if any of the following properties are not satisfied.

1. vk is of the form $\left(\Pi, g, h, g_0, (g^{\mathcal{P}(L_i)}, g^{\mathcal{P}(R_i)})_{i \in [\eta]}\right)$.
2. $\mathsf{GrpVfy}(\Pi) = 1$ and $\mathsf{GrpVfy}(\Pi, s) = 1$ for all $s \in (g, h, g_0) \cup (g^{\mathcal{P}(L_i)}, g^{\mathcal{P}(R_i)})_{i \in [\eta]}$.

Then, it checks the validity of X, Y and π. In doing so, it first computes the coefficients $(\alpha_S)_{S \subseteq [\zeta]}$ of the multi-variate polynomial

$$p(\mathsf{Z}_1, \cdots, \mathsf{Z}_\zeta) = \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \left((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot \mathsf{Z}_k\right) = \sum_{S \subseteq [\zeta]} \alpha_S \prod_{k \in S} \mathsf{Z}_k.$$

Next, for all $i \in [\eta]$ and $S \subseteq [\zeta]$, it sets $L_S = S \cap [\lfloor \zeta/2 \rfloor]$ and $R_S = S \cap [\lfloor \zeta/2 \rfloor + 1, \zeta]$, and computes $\Phi_{i,S}$ as

$$\Phi_{i,S} = e(g^{\prod_{k \in L_S} w_{i,k}}, g^{\prod_{k \in R_S} w_{i,k}}).$$

Here, in case $L_S = \phi$ (resp. $R_S = \phi$), we define $\prod_{k \in L_S} w_{i,k}$ (resp. $\prod_{k \in R_S} w_{i,k}$) to be 1. Note that these values can be computed efficiently, since $g^{\mathcal{P}(L_i)}, g^{\mathcal{P}(R_i)}$ are given as part of the verification key. It outputs 0 if any of the following properties are not satisfied.

3. $X \in \{0,1\}^n, Y \in \mathbb{G}_T$, π is of the form $\pi = (\pi_0, (\pi_i)_{i\in[\eta]}, (\pi_{[1:i']})_{i'\in[2,\eta]})$.
4. It holds that for all $i \in [\eta]$ and $i' \in [3,\eta]$,

$$e(\pi_i, g) = \prod_{S\subseteq[\zeta]} \Phi_{i,S}^{\alpha_S}, \quad \text{and} \quad e(\pi_{[1:2]}, g) = e(\pi_1, \pi_2), \quad \text{and} \quad e(\pi_{[1:i']}, g) = e(\pi_{[1:i'-1]}, \pi_{i'}).$$

5. It holds that $e(\pi_{[1:\eta]}, g) = e(\pi_0, g_0)$ and $e(\pi_0, h) = Y$.

If all the above checks are passed, it outputs 1.

The correctness, unique provability and pseudorandomness of the above VRF can be proven in a similar manner to the VRF in Sec. 5.2. For completeness, we provide some details in Appendix C. Finally, we end this section by discussing the efficiency of the above construction.

Remark 3. *Our second VRF has verification key size* $|\mathsf{vk}| = \omega(\sqrt{\lambda}\log\lambda)$ *and proof size* $|\pi| = \omega(\log\lambda)$. *To see this, observe that for all* $i \in [\eta]$ *we have* $|\mathcal{P}(L_i)|, |\mathcal{P}(R_i)| \le 2^{\lceil\zeta/2\rceil}$, *which follows from* $|L_i|, |R_i| \le \lceil\zeta/2\rceil$. *Next, since* $\ell = \Theta(\lambda)$, *there exists some positive constant* c *such that* $\ell(\lambda) \le c\lambda$ *for large enough* $\lambda \in \mathbb{N}$. *Then, since* $\zeta = \lfloor\log\ell\rfloor + 1$, *we have* $\zeta(\lambda) \le \log\lambda + \log c + 1$. *Therefore,* $2^{\lceil\zeta/2\rceil} \le 2^{\zeta/2+1} = c'\lambda^{1/2}$ *for some positive constant* c'. *Thus, we obtain the upper bound* $|g^{\mathcal{P}(L_i)}|, |g^{\mathcal{P}(R_i)}| = O(\sqrt{\lambda})$. *Since we consider this for all* $i \in [\eta]$ *where* $\eta = \omega(\log\lambda)$, *we conclude* $|\mathsf{vk}| = \omega(\sqrt{\lambda}\log\lambda)$. *Note that this means that we can take* vk *for example as small as* $|\mathsf{vk}| = O(\sqrt{\lambda}\log^2\lambda)$. *A detailed comparison is provided in Sec. 1.1, Table 1.*

6 Predicate Encryption for MultD-Eq Predicates

In this section, we show how to construct a predicate encryption scheme for the multi-dimensional equality predicates MultD-Eq. This directly yields predicate encryption schemes for all the predicates presented in Sec. 4.3. Due to the symmetry of the MultD-Eq predicate and the compatible domains $(\mathcal{X}, \mathcal{Y})$, we obtain both key-policy and ciphertext-policy predicate encryption schemes.

6.1 Embedding Predicate Encoding Schemes into Matrices

[BGG+14] provides us with a generic way of constructing a lattice-based attribute-based encryption (ABE) scheme from three deterministic algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$. In this paper, we slightly modify the syntax of the $\mathsf{Eval}_{\mathrm{ct}}$ algorithm so that the three deterministic algorithms yield a predicate encryption (equivalently, a predicate hiding ABE) scheme.

Definition 12. *We say that the* deterministic *algorithms* $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\text{ct-priv}}, \mathsf{Eval}_{\mathrm{sim}})$ *are* $\alpha_{\mathcal{C}}$*-predicate encryption (PE) enabling for a family of arithmetic circuits* $\mathcal{C} = \{C : \mathbb{Z}_q^t \to \mathbb{Z}_q\}$ *if they are efficient and satisfy the following properties:*

- $\mathsf{Eval}_{\mathrm{pk}}\left(C \in \mathcal{C}, \quad \mathbf{B}_0, (\mathbf{B}_i)_{i\in[t]} \in \mathbb{Z}_q^{n\times m}\right) \to \mathbf{B}_C \in \mathbb{Z}_q^{n\times m}$
- $\mathsf{Eval}_{\text{ct-priv}}\left(C \in \mathcal{C}, \quad \mathbf{c}_0, (\mathbf{c}_i)_{i\in[t]} \in \mathbb{Z}_q^{n}\right) \to \mathbf{c}_C \in \mathbb{Z}_q^{m}$
- $\mathsf{Eval}_{\mathrm{sim}}\left(C \in \mathcal{C}, \quad \mathbf{R}_0, (\mathbf{R}_i)_{i\in[t]} \in \mathbb{Z}^{m\times m}\right) \to \mathbf{R}_C \in \mathbb{Z}^{m\times m}$

We further require that the following holds:

1. $\mathsf{Eval}_{\mathrm{pk}}(C, (\mathbf{A}\mathbf{R}_0 - \mathbf{G}), (\mathbf{A}\mathbf{R}_i - x_i\mathbf{G})_{i\in[t]}) = \mathbf{A} \cdot \mathsf{Eval}_{\mathrm{sim}}(C, \mathbf{R}_0, (\mathbf{R}_i)_{i\in[t]}) - C(\mathbf{x})\mathbf{G}$ *for any* $\mathbf{x} = (x_1, \cdots, x_t) \in \{0,1\}^t$.

2. *If* $\mathbf{c}_0 = (\mathbf{B}_0 + \mathbf{G})^\top \mathbf{s} + \mathbf{z}_0$ *and* $\mathbf{c}_i = (\mathbf{B}_i + x_i\mathbf{G})^\top \mathbf{s} + \mathbf{z}_i$ *for some* $\mathbf{s} \in \mathbb{Z}_q^n$, *and* $\mathbf{z}_0, \mathbf{z}_i \leftarrow D_{\mathbb{Z}^m,\beta}, x_i \in \{0,1\}$ *for all* $i \in [t]$, *then* $\|\mathbf{c}_C - (\mathbf{B}_C + C(\mathbf{x})\mathbf{G})^\top \mathbf{s}\|_2 < \alpha_{\mathcal{C}} \cdot \beta\sqrt{m}$ *with all but negligible probability.*

3. *If* $\mathbf{R}_i \leftarrow \{-1,1\}^{m\times m}$ *for all* $i \in [0,t]$, *then* $s_1(\mathbf{R}_C) < \alpha_{\mathcal{C}}$ *with all but negligible probability.*

There are two major differences between the notions from [BGG+14]. First, $\mathsf{Eval}_{\text{ct-priv}}$ does not take $(x_i)_{i\in[t]} \in \{0,1\}^t$ as input to the homomorphic evaluation of the ciphertexts. On one hand this limits us to perform only linear operations over the ciphertexts, however, on the other hand this will allow the decryptor to create $\mathbf{c}_C$ without knowledge of the predicate associated to the ciphertext (See also [AFV11]). Second, we loosen the condition on $\mathbf{z}, \mathbf{R}$ in Requirement 1, 2 to hold with overwhelming probability. This allows us to obtain tighter bounds on the behavior of the random matrices and error vectors. Finally, we make a minor change by additionally including $(\mathbf{B}_0, \mathbf{c}_0, \mathbf{R}_0)$ as inputs to the algorithms to cope with the constant terms of the polynomials being evaluated.

$\alpha_{\mathcal{C}}$-PE enabling algorithms for MultD-Eq predicates. We show that the linear predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}}$ for the MultD-Eq predicates (Sec. 4.2, Lem. 9) provides us with a family of arithmetic circuits $\hat{\mathcal{C}}$ that allows for $\alpha_{\hat{\mathcal{C}}}$-PE enabling algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\text{ct-priv}}, \mathsf{Eval}_{\mathrm{sim}})$. Let all the parameters be defined as in Lem. 9 and denote $\hat{\mathcal{C}}$ as the set $\{\hat{C}_{\mathsf{Y}} \mid \hat{C}_{\mathsf{Y}} \leftarrow \mathsf{EncPred}_{\mathsf{Lin}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}), \forall \mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}\}$. The three algorithms are defined as follows:[15]

$\mathsf{Eval}_{\mathrm{pk}}\big(\hat{C}_{\mathsf{Y}} \in \hat{\mathcal{C}},\ \mathbf{B}_0, (\mathbf{B}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\big)$: It outputs

$$\mathbf{B}_{\mathsf{Y}} = \mathbf{B}_0 \cdot \mathbf{G}^{-1}(D\mathbf{G}) - \sum_{i=1}^{D}\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \mathbf{B}_{i,j,w} \in \mathbb{Z}_q^{n\times m}.$$

$\mathsf{Eval}_{\text{ct-priv}}\Big(\hat{C}_{\mathsf{Y}} \in \hat{\mathcal{C}},\ \mathbf{c}_0, (\mathbf{c}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\Big)$: It outputs

$$\mathbf{c} = \big(\mathbf{G}^{-1}(D\mathbf{G})\big)^\top \mathbf{c}_0 - \sum_{i=1}^{D}\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \mathbf{c}_{i,j,w} \in \mathbb{Z}_q^{m}.$$

$\mathsf{Eval}_{\mathrm{sim}}\Big(\hat{C}_{\mathsf{Y}} \in \hat{\mathcal{C}},\ \mathbf{R}_0, (\mathbf{R}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\Big)$: It outputs

$$\mathbf{R}_{\mathsf{Y}} = \mathbf{R}_0 \cdot \mathbf{G}^{-1}(D\mathbf{G}) - \sum_{i=1}^{D}\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \mathbf{R}_{i,j,w} \in \mathbb{Z}_q^{n\times m}.$$

Lemma 12. *The above algorithms* $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\text{ct-priv}}, \mathsf{Eval}_{\mathrm{sim}})$ *are* $\alpha_{\hat{\mathcal{C}}}$*-PE enabling algorithms for the family of arithmetic circuits* $\hat{\mathcal{C}}$ *defined by the predicate encoding scheme* $\mathsf{PES}_{\mathsf{Lin}}$ *for the* MultD-Eq *predicates defined over* $\mathbb{Z}_p^{D\times\ell}$, *where* $\alpha_{\hat{\mathcal{C}}} = C \cdot \max\{m\sqrt{m/n}, \sqrt{D\ell pm}\}$ *for some absolute constant* $C > 0$.

[15] Recall that when we use the notation $(A_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}$, we assume the elements are sorted in the lexicographical order.

Proof. We check that all the requirements of Def. 12 are satisfied. First, by the property of the gadget matrix $\mathbf{G}$ and the fact that $a_{i,j,w} \in \{-1, 0, 1\}$, we have $a_{i,j,w}\mathbf{I}_m = \mathbf{G}^{-1}(a_{i,j,w} \cdot \mathbf{G})$. Then, by plugging in $\mathbf{B}_0 = \mathbf{A}\mathbf{R}_0 - \mathbf{G}$ and $\mathbf{B}_{i,j,w} = \mathbf{A}\mathbf{R}_{i,j,w} - \hat{\mathsf{X}}_{i,j,w}\mathbf{G}$, we can see that $\mathbf{B}_{\mathsf{Y}} = \mathbf{A}\mathbf{R}_{\mathsf{Y}} - \hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}})\mathbf{G}$. Hence, Requirement 1 holds. Furthermore, simple calculation shows that in case $\mathbf{c}_0 = (\mathbf{B}_0 + \mathbf{G})^\top \mathbf{s} + \mathbf{z}_0$ and $\mathbf{c}_{i,j,w} = (\mathbf{B}_{i,j,w} + \hat{\mathsf{X}}_{i,j,w}\mathbf{G})^\top \mathbf{s} + \mathbf{z}_{i,j,w}$, we have

$$\mathbf{c} = (\mathbf{B}_{\mathsf{Y}} + \hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}})\mathbf{G})^\top \mathbf{s} + \Big(\underbrace{\big(\mathbf{G}^{-1}(D\mathbf{G})\big)^\top \mathbf{z}_0}_{:=\mathbf{e}_1 \text{ (noise)}} - \underbrace{\sum_{i=1}^{D}\sum_{j=1}^{\ell}\sum_{w=1}^{L} a_{i,j,w} \cdot \mathbf{z}_{i,j,w}}_{:=\mathbf{e}_2 \text{ (noise)}} \Big). \tag{20}$$

Recall that the discrete Gaussian distribution $D_{\mathbb{Z}^m,\beta}$ is subgaussian with parameter $C \cdot \beta$ for some absolute constant C. In the following, with an abuse of notation, we will denote any absolute constant as C. Then by the property of $\mathbf{G}^{-1}$, we can use Lem. 2 with $B = C\beta$ and $\ell = 1$ to obtain $\|\mathbf{e}_1\|_2 \le C \cdot \beta m\sqrt{m/n}$. Note that we assume $n|m$ without loss of generality. Next, from Lem. 4 and the linearity of subgaussian variables, we have $\|\mathbf{e}_2\|_2 \le C \cdot \sqrt{mD\ell L}\beta$. Combining this together with the fact $L = 2^{\lfloor \log p \rfloor + 1}$, we obtain $\|\mathbf{e}_1 - \mathbf{e}_2\|_2 \le C \cdot (m/\sqrt{n} + \sqrt{D\ell p}) \cdot \sqrt{m}\beta \le \alpha_{\hat{C}} \cdot \sqrt{m}\beta$ with all but negligible probability. This shows that Requirement 2 holds.

Finally, we show that Requirement 3 holds. First, since the absolute values of each element is bounded by 1, every entry of $\mathbf{R}_0, \mathbf{R}_{i,j,w}$ are subgaussian variables with parameter C. Then, following a similar argument as above, $\mathbf{R}_{\mathsf{Y}}$ is a subgaussian matrix with parameter $C \cdot (m/\sqrt{n} + \sqrt{D\ell p}) \cdot \sqrt{m}$. Then using the Lem. 2.9 of [MP12], we have that $s_1(\mathbf{R}_{\mathsf{Y}}) \le C \cdot (m\sqrt{m/n} + \sqrt{D\ell pm}) \le C \cdot \max\{m\sqrt{m/n}, \sqrt{D\ell pm}\} \le \alpha_{\hat{C}}$ with all but negligible probability. □

6.2 Construction

Given $\alpha_{\hat{\mathcal{C}}}$-PE enabling algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\text{ct-priv}}, \mathsf{Eval}_{\mathrm{sim}})$ for a family of arithmetic circuits defined by the predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}} = (\mathsf{EncInpt}_{\mathsf{Lin}}, \mathsf{EncPred}_{\mathsf{Lin}})$ for the MultD-Eq predicates with compatible domains $(\mathcal{X}, \mathcal{Y})$, we build a predicate encryption scheme for the same family of predicates.

Parameters. In the following, let n, m, q, p, D, ℓ be positive integers such that q is a prime and $q > D$, and let σ, α, α' be positive reals denoting the Gaussian parameters. Furthermore, let $(\mathcal{X}, \mathcal{Y}) \in \mathbb{Z}_p^{D\times\ell} \times \mathbb{Z}_p^{D\times\ell}$ be any compatible domains for the MultD-Eq predicates, let $\mathcal{P} = \{\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} : \mathcal{X} \to \{0,1\} \mid \mathsf{Y} \in \mathcal{Y}\}$ be the set of multi-dimensional predicates and $\hat{\mathcal{C}} = \{\hat{C}_{\mathsf{Y}} \mid \hat{C}_{\mathsf{Y}} \leftarrow \mathsf{EncPred}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}), \forall \mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}\}$ be the set of polynomials representing the multi-dimensional predicates. Finally, let $\zeta = \lfloor \log p \rfloor + 1$ and $L = 2^\zeta$. Here, we assume that all of the parameters are a function of the security parameter $\lambda \in \mathbb{N}$. We provide a concrete parameter selection of the scheme in Sec. 6.3. The following is our PE scheme.

$\mathsf{Setup}(1^\lambda)$: It first runs $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ to obtain $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m\times m}$. It also picks $\mathbf{u} \leftarrow \mathbb{Z}_q^n$, $\mathbf{B}_0, \mathbf{B}_{i,j,w} \leftarrow \mathbb{Z}_q^{n\times m}$ for $(i,j,w) \in [D] \times [\ell] \times [L]$ and outputs

$$\mathsf{mpk} = \Big(\mathbf{A}, \mathbf{B}_0, (\mathbf{B}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}, \mathbf{u}\Big) \quad \text{and} \quad \mathsf{msk} = \mathbf{T}_{\mathbf{A}}.$$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{MultD\text{-}Eq}_{\mathsf{Y}})$: Given a predicate $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}$ for $\mathsf{Y} \in \mathbb{Z}_p^{D\times\ell}$ as input, it runs $\hat{C}_{\mathsf{Y}} \leftarrow \mathsf{EncPred}_{\mathsf{Lin}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}})$ and computes

$$\mathsf{Eval}_{\mathrm{pk}}\Big(\hat{C}_{\mathsf{Y}}, \mathbf{B}_0, (\mathbf{B}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\Big) \to \mathbf{B}_{\mathsf{Y}} \in \mathbb{Z}_q^{n\times m}.$$

Then, it runs

$$\mathbf{e} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}_{\mathsf{Y}}, \mathbf{u}, \mathbf{T}_{\mathbf{A}}, \sigma),$$

where $[\mathbf{A}|\mathbf{B}_{\mathsf{Y}}]\mathbf{e} = \mathbf{u} \mod q$, and finally returns $\mathsf{sk}_{\mathsf{Y}} = \mathbf{e} \in \mathbb{Z}^{2m}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{X}, \mathsf{M})$: Given an attribute $\mathsf{X} \in \mathbb{Z}_p^{D\times\ell}$ as input, it first runs $\hat{\mathsf{X}} \leftarrow \mathsf{EncInpt}_{\mathsf{Lin}}(\mathsf{X})$ where $\hat{\mathsf{X}} \in \{0,1\}^{D\ell L}$. Then it samples $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $z \leftarrow D_{\mathbb{Z},\alpha q}$, $\mathbf{z}, \mathbf{z}_0, \mathbf{z}_{i,j,w} \leftarrow D_{\mathbb{Z}^m,\alpha' q}$ for $(i,j,w) \in [D]\times[\ell]\times[L]$, and computes

$$\mathbf{c}_{\mathsf{X}} = \begin{cases} c & = \mathbf{u}^\top\mathbf{s} + z + \mathsf{M}\cdot\lfloor q/2\rceil, \\ \mathbf{c} & = \mathbf{A}^\top\mathbf{s} + \mathbf{z}, \\ \mathbf{c}_0 & = (\mathbf{B}_0 + \mathbf{G})^\top\mathbf{s} + \mathbf{z}_0, \\ \mathbf{c}_{i,j,w} & = \left(\mathbf{B}_{i,j,w} + \hat{\mathsf{X}}_{i,j,w}\mathbf{G}\right)^\top\mathbf{s} + \mathbf{z}_{i,j,w} \quad \text{for} \quad (i,j,w) \in [D]\times[\ell]\times[L], \end{cases}$$

where $\hat{\mathsf{X}}_{i,j,w}$ is the (i,j,w)-th element of $\hat{\mathsf{X}}$. Finally, it returns the ciphertext $\mathbf{c}_{\mathsf{X}} \in \mathbb{Z}_q \times (\mathbb{Z}_q^m)^{D\ell L+2}$.

$\mathsf{Dec}(\mathsf{mpk}, (\hat{C}_{\mathsf{Y}}, \mathsf{sk}_{\mathsf{Y}}), \mathbf{c}_{\mathsf{X}})$: To decrypt the ciphertext $\mathbf{c}_{\mathsf{X}} = (c, \mathbf{c}, \mathbf{c}_0, (\mathbf{c}_{i,j,w}))$ given a predicate and a secret key $(\hat{C}_{\mathsf{Y}}, \mathsf{sk}_{\mathsf{Y}})$, it computes

$$\mathsf{Eval}_{\text{ct-priv}}\left(\hat{C}_{\mathsf{Y}}, \mathbf{c}_0, (\mathbf{c}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\right) \to \bar{\mathbf{c}} \in \mathbb{Z}_q^m.$$

Then using the secret key $\mathsf{sk}_{\mathsf{Y}} = \mathbf{e} \in \mathbb{Z}^{2m}$, it computes

$$d = c - [\mathbf{c}^\top|\bar{\mathbf{c}}^\top]^\top\mathbf{e} \in \mathbb{Z}_q.$$

Finally, it returns $|d - \lfloor q/2\rceil| < q/4$ and 0 otherwise.

6.3 Correctness and Parameter Selection

Lemma 13 (correctness)**.** *If the predicate is satisfied, assuming $\alpha' > \alpha$, the error term on the decrypted values are bounded by $O(\sqrt{m}\alpha'\alpha_{\hat{\mathcal{C}}}\sigma q)$ with overwhelming probability.*

Proof. By the definition of $\alpha_{\hat{\mathcal{C}}}$-PE enabling algorithms, when the cryptosystem is operated as specified, we have during decryption

$$d = c - [\mathbf{c}^\top|\bar{\mathbf{c}}^\top]^\top\mathbf{e} = \mathsf{M}\cdot\lfloor q/2\rceil + z - (\mathbf{z}_0 + \mathbf{z})^\top\mathbf{e},$$

where $\mathbf{z}$ is defined as in Eq. (20) (i.e., $\mathbf{z} := \mathbf{e}_1 - \mathbf{e}_2$). Further, we have the following upper bound on the noise.

$$\begin{aligned} \|z - (\mathbf{z}_0 + \mathbf{z})^\top\mathbf{e}\|_2 &\le |z| + (\|\mathbf{z}_0^\top\mathbf{e}\|_2 + \|\mathbf{z}^\top\mathbf{e}\|_2) \\ &\le \sqrt{m}\alpha q + 2\sqrt{m}\alpha'\alpha_{\hat{\mathcal{C}}}\sigma q \\ &= O(\sqrt{m}\alpha'\alpha_{\hat{\mathcal{C}}}\sigma q). \end{aligned}$$

The first inequality follows from the CauchySchwarz inequality and the second inequality follows from Lem. 4, Requirement 2 of the $\alpha_{\hat{\mathcal{C}}}$-PE enabling algorithms and the linearity of subgaussian variables. □

Parameter selection. To satisfy the correctness requirement and make the security proof follow through, we need the following:

- the error term is less than $q/5$ with overwhelming probability (i.e., $O(\sqrt{m}\alpha'\alpha_{\hat{C}}\sigma q) < q/5$. See Lem. 13),
- the correctness of $\mathsf{PES}_{\mathsf{Lin}}$ holds. (i.e., $q > D$. See Lem. 9),
- the $\mathsf{TrapGen}$ algorithm works as specified during Setup. (i.e., $m \geq 2n\lceil \log q \rceil$. See Lem. 6),
- the leftover hash lemma can be applied in the security proof (i.e., $m > (n+1)\log q + \omega(\log n)$. See. Lem. 3),
- the $\mathsf{SampleLeft}$ algorithm works as specified during KeyGen. (i.e., $\sigma > \|\mathbf{T}_{\mathbf{A}}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$. See Lem. 6),
- the $\mathsf{SampleRight}$ algorithm in the security proof works as specified. (i.e., $\sigma > s_1(\mathbf{R}_{\mathsf{Y}}) \cdot \|\mathbf{T}_{\mathbf{G}}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m}) \Leftrightarrow \sigma > \alpha_{\hat{C}} \cdot \omega(\sqrt{\log m})$. See Lem. 6, Def. 12),
- the ReRand algorithm in the security proof works as specified (i.e., $\alpha'/2\alpha > s_1(\mathbf{R}^*)$, $\alpha q > \omega(\sqrt{\log mD\ell L})$ where $\mathbf{R}^*$ is defined as in Game_4. See Lem. 5),
- the worst case to average case reduction works (i.e., $\alpha q > 2\sqrt{n}$).

To satisfy the above requirements, one way to set the parameters are as follows:

$$m = O(n \log q), \qquad q = \sqrt{m} \cdot (\sqrt{D\ell p})^{-1} \cdot \alpha_{\hat{C}}^2 \cdot \omega(\log m), \qquad \sigma = \alpha_{\hat{C}} \cdot \omega(\sqrt{\log m}),$$
$$\alpha_{\hat{C}} = O(\max\{m\sqrt{\log q}, \sqrt{D\ell pm}\}) \qquad \alpha = (\sqrt{D\ell p}) \cdot \alpha_{\hat{C}}^{-2} \cdot \omega(\log m)^{-1}, \qquad \alpha' = O(\sqrt{D\ell pm} \cdot \alpha),$$

and round up q to the nearest larger prime. Here, D, ℓ, p are chosen accordingly to the types of $\mathsf{MultD\text{-}Eq}$ predicates one wants to use.

6.4 Security Proof

Theorem 4. *Given the PE enabling algorithms* $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\text{ct-priv}}, \mathsf{Eval}_{\mathrm{sim}})$ *for the family of arithmetic circuits* $\hat{\mathcal{C}}$ *defined above, our predicate encryption scheme is selectively secure and weakly attribute hiding with respect to the* $\mathsf{MultD\text{-}Eq}$ *predicates, assuming the hardness of* $\mathsf{LWE}_{n,m+1,q,D_{\mathbb{Z},\alpha q}}$.

Proof. The proof proceeds in a sequence of games where the first game is identical to the real predicate encryption security game from Def. 2. In the last game in the sequence, the adversary has zero advantage. In the following, let $\mathcal{A}$ be a PPT adversary that breaks the security of the scheme with advantage ϵ, and define E_i to be the event that $\mathcal{A}$ wins in Game_i.

Game_0 : This is the real security game between the attacker $\mathcal{A}$ against our scheme. By definition, we have $|\Pr[\mathsf{E}_0] - 1/2| = \epsilon$. In the following, let $\mathsf{X}^* \in \mathbb{Z}_p^{D\times\ell}$ denote the challenge attribute $\mathcal{A}$ submits.

Game_1 : In this game, we change the way $\mathbf{B}_0, \mathbf{B}_{i,j,w}$ are chosen. At the beginning of the game, the challenger samples $\mathbf{R}_0, \mathbf{R}_{i,j,w} \leftarrow \{-1,1\}^{m\times m}$ for $(i,j,w) \in [D] \times [\ell] \times [L]$. Then, we define $\mathbf{B}_0$ and $\mathbf{B}_{i,j,w}$ as

$$\mathbf{B}_0 = \mathbf{A}\mathbf{R}_0 - \mathbf{G} \quad \text{and} \quad \mathbf{B}_{i,j,w} = \mathbf{A}\mathbf{R}_{i,j,w} - \hat{\mathsf{X}}^*_{i,j,w}\mathbf{G}, \tag{21}$$

where $\hat{\mathsf{X}}^* \leftarrow \mathsf{EncInpt}_{\mathsf{Lin}}(\mathsf{X}^*)$ and $\hat{\mathsf{X}}^*_{i,j,w}$ is the (i,j,w)-th element of $\hat{\mathsf{X}}^* \in \{0,1\}^{D\ell L}$. By Lem. 3, the distributions

$$\left(\mathbf{A}, \mathbf{B}_0, (\mathbf{B}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\right) \quad \text{and} \quad \left(\mathbf{A}, \mathbf{A}\mathbf{R}_0, (\mathbf{A}\mathbf{R}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\right)$$

are negligibly close, where $\mathbf{B}_0, \mathbf{B}_{i,j,w} \leftarrow \mathbb{Z}_q^{n\times m}$. Therefore, we have

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda).$$

Before continuing to our next game, we make the following observation. From Requirement 1 of Def. 12, for all $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}$ and $\hat{C}_{\mathsf{Y}} \leftarrow \mathsf{EncPred}_{\mathsf{Lin}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}})$, we have

$$\mathsf{Eval}_{\mathrm{pk}}\left(\hat{C}_{\mathsf{Y}}, (\mathbf{A}\mathbf{R}_0 - \mathbf{G}), (\mathbf{A}\mathbf{R}_{i,j,w} - \hat{\mathsf{X}}^*_{i,j,w}\mathbf{G})_{(i,j,w)}\right) = \mathbf{A}\mathbf{R}_{\mathsf{Y}} - \hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}}^*)\mathbf{G}$$

where $\mathbf{R}_{\mathsf{Y}} = \mathsf{Eval}_{\mathrm{sim}}(\hat{C}_{\mathsf{Y}}, \mathbf{R}_0, (\mathbf{R}_{i,j,w})_{(i,j,w)})$. Furthermore, we have $\|\mathbf{R}_{\mathsf{Y}}\|_2 < \alpha_{\hat{C}} < \sigma$ from Requirement 3 and our parameter selection. Now by the correctness of the $\mathsf{PES}_{\mathsf{Lin}}$ scheme (Lem. 9), we have

$$\mathbf{A}\mathbf{R}_{\mathsf{Y}} - \hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}})\mathbf{G} = \begin{cases} \mathbf{A}\mathbf{R}_{\mathsf{Y}} & \text{if} \quad \mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}^*) = 1 \\ \mathbf{A}\mathbf{R}_{\mathsf{Y}} - t\mathbf{G} \quad \text{for} \ \exists t \in \{1, \cdots D\} & \text{if} \quad \mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}^*) = 0 \end{cases}. \tag{22}$$

Note that since $q > D$ and q a prime, t is always invertible in $\mathbb{Z}_q$.

Game_2 : In this game, we change how $\mathbf{A}$ is sampled. Namely, in Game_2, we generate $\mathbf{A}$ as a random matrix in $\mathbb{Z}_q^{n\times m}$ instead of generating it with a trapdoor. By Lem. 6, this makes only negligible difference. To respond to a key extraction query for a predicate $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}} \in \mathcal{P}$ made by $\mathcal{A}$, it first runs $\hat{C}_{\mathsf{Y}} \leftarrow \mathsf{EncPred}_{\mathsf{Lin}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}})$ and computes

$$\mathsf{Eval}_{\mathrm{sim}}\left(\hat{C}_{\mathsf{Y}}, \mathbf{R}_0, (\mathbf{R}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\right) \to \mathbf{R}_{\mathsf{Y}}.$$

If $\mathcal{A}$ is a valid adversary, then all the predicates submitted as a key extraction query satisfies $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}(\mathsf{X}^*) = 0$, and by Eq. (22) we have $t = \hat{C}_{\mathsf{Y}}(\hat{\mathsf{X}}^*)$ for some invertible element $t \in \{1, \cdots, D\} \subset \mathbb{Z}_q$. Then, using the $\mathsf{SampleRight}$ algorithm from Lem. 6, it samples the secret key

$$\mathsf{SampleRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}_{\mathsf{Y}}, t, \mathbf{u}, \mathbf{T}_{\mathbf{G}}, \sigma) \to \mathbf{e}.$$

Note that in the previous game the key was sampled as

$$\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}_{\mathsf{Y}}, \mathbf{u}, \mathbf{T}_{\mathbf{A}}, \sigma) \to \mathbf{e}.$$

By the definition of $\mathsf{Eval}_{\mathrm{sim}}, \mathsf{SampleRight}, \mathsf{SampleLeft}$ and for our choice of σ, the distribution of the secret key is negligibly close to the distribution of that in the previous game. Therefore, the above alters the view of $\mathcal{A}$ only negligibly. Thus, we have

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda).$$

Game_3 : In this game, we change the way the challenge ciphertext is created when $\mathsf{coin} = 0$. Recall in the previous games when $\mathsf{coin} = 0$, we created a valid challenge ciphertext as in the real scheme. If $\mathsf{coin} = 0$, to create the challenge ciphertext, the challenger first picks $\mathbf{s} \leftarrow \mathbb{Z}_q^n, z \leftarrow D_{\mathbb{Z},\alpha q}, \bar{\mathbf{z}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ and computes $v = \mathbf{u}^\top \mathbf{s} + z \in \mathbb{Z}_q$ and $\mathbf{v} = \mathbf{A}^\top \mathbf{s} + \bar{\mathbf{z}} \in \mathbb{Z}_q^m$. It then sets $\mathbf{R}^* = [\mathbf{R}_0 | \mathbf{R}_{1,1,1} | \cdots | \mathbf{R}_{D,\ell,L}] \in \mathbb{Z}^{m \times (D\ell L+1)m}$ and runs

$$\mathsf{ReRand}\left([\mathbf{I}_m | \mathbf{R}^*], \mathbf{v}, \alpha q, \frac{\alpha'}{2\alpha}\right) \to \mathbf{c}' \in \mathbb{Z}_q^{(D\ell L+2)m}$$

from Lem. 5, where $\mathbf{I}_m$ is the identity matrix with size m. Finally, it parses $\mathbf{c}'$ appropriately into $D\ell L + 2$ size m vectors $(\mathbf{c}, \mathbf{c}_0, (\mathbf{c}_{i,j,w})_{(i,j,w)})$ and outputs the challenge ciphertext as

$$\left(c = v + \mathsf{M} \cdot \lfloor q/2 \rceil, \quad \mathbf{c}, \quad \mathbf{c}_0, \quad (\mathbf{c}_{i,j,w})_{(i,j,w)\in[D]\times[\ell]\times[L]}\right). \tag{23}$$

We claim this change alters the view of $\mathcal{A}$ only negligibly. To see this, we apply the noise rerandomization lemma (Lem. 5) with $\mathbf{V} = [\mathbf{I}_m | \mathbf{R}^*], \mathbf{b} = \mathbf{A}^\top \mathbf{s}$ and $\mathbf{z} = \bar{\mathbf{z}}$ to obtain that the distribution of $\mathbf{c}'$ is negligibly close to the following:

$$\begin{aligned}
\mathbf{c}'^\top &= \mathbf{s}^\top \mathbf{A}[\mathbf{I}_m | \mathbf{R}^*] + \mathbf{z}'^\top \\
&= \mathbf{s}^\top [\mathbf{A} | \mathbf{A}\mathbf{R}_0 | \mathbf{A}\mathbf{R}_{1,1,1} | \cdots | \mathbf{A}\mathbf{R}_{D,\ell,L}] + \mathbf{z}'^\top \\
&= \mathbf{s}^\top [\mathbf{A} | \mathbf{B}_0 + \mathbf{G} | \mathbf{B}_{1,1,1} + \hat{\mathsf{X}}^*_{1,1,1}\mathbf{G} | \cdots | \mathbf{B}_{D,\ell,L} + \hat{\mathsf{X}}^*_{D,\ell,L}\mathbf{G}] + \mathbf{z}'^\top \in \mathbb{Z}^{(D\ell L+2)m}
\end{aligned}$$

where the last equality follows from Eq. (21), and $\mathbf{z}'$ is distributed negligibly close to $D_{\mathbb{Z}^{(D\ell L+2)m},\alpha' q}$. Here, we can apply the noise rerandomization lemma since

$$\alpha'/2\alpha > 20\sqrt{(D\ell L + 3)m} \geq s_1([\mathbf{I}_m | \mathbf{R}^*]),$$

for our parameter selection, where we use Lem. 1 for the second inequality. It can be seen that the challenge ciphertext is distributed statistically close to Game_2. Therefore, we may conclude that

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\lambda).$$

Game_4 : In this game, we further change the way the challenge ciphertext is created when $\mathsf{coin} = 0$. If $\mathsf{coin} = 0$, to create the challenge ciphertext the challenger first picks $w \leftarrow \mathbb{Z}_q, \mathbf{w} \leftarrow \mathbb{Z}_q^m$, $z \leftarrow D_{\mathbb{Z},\alpha q}$, and $\bar{\mathbf{z}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ and computes $v = w + z \in \mathbb{Z}_q$ and $\mathbf{v} = \mathbf{w} + \bar{\mathbf{z}} \in \mathbb{Z}_q^m$. It then sets $\mathbf{R}^*$ and runs the ReRand algorithm as in Game_4. Finally, it sets the challenge ciphertext as in Eq. (23). We claim that $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]|$ is negligible assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,D_{\mathbb{Z},\alpha q}}$ problem. To show this, we use $\mathcal{A}$ to construct an LWE adversary $\mathcal{B}$ as follows:

$\mathcal{B}$ is given the problem instance of LWE as $(\mathbf{A}', \mathbf{v}' = \mathbf{w}' + \bar{\mathbf{z}}') \in \mathbb{Z}_q^{n\times(m+1)} \times \mathbb{Z}_q^{m+1}$ where $\bar{\mathbf{z}}' \leftarrow D_{\mathbb{Z}^{m+1},\alpha q}$. The task of $\mathcal{B}$ is to distinguish whether $\mathbf{w}' = \mathbf{A}'^\top \mathbf{s}$ for $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ or $\mathbf{w}' \leftarrow \mathbb{Z}_q^{m+1}$. In the following, let the first column of $\mathbf{A}'$ be $\mathbf{u} \in \mathbb{Z}_q^n$ and the remaining columns be $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. Further, let the first coefficient of $\mathbf{v}'$ be v and the remaining coefficients be $\mathbf{v} \in \mathbb{Z}_q^m$. Using these terms, $\mathcal{B}$ sets the master public keys as in Eq. (21). During the game, key extractions queries made by $\mathcal{A}$ are answered as in Game_2 without knowledge of the trapdoor of $\mathbf{A}$. To generate the challenge ciphertext, it first picks $\mathsf{coin} \leftarrow \{0, 1\}$. If $\mathsf{coin} = 0$,

it generates the challenge ciphertext as in Eq. (23) using $v, \mathbf{v}$, and returns it to $\mathcal{A}$. Note that all $\mathcal{B}$ needs to do to generate the ciphertext is to run the ReRand algorithm, which it can do without knowledge of the secret randomness $\mathbf{s}, \bar{\mathbf{z}}'$. If $\mathsf{coin} = 1$, $\mathcal{B}$ returns a random ciphertext. At the end of the game, coin' is defined. Finally, $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$ and 0 otherwise. It can be seen that if $\mathbf{A}', \mathbf{v}'$ is a valid LWE sample (i.e., $\mathbf{v}' = \mathbf{A}'^{\top}\mathbf{s}$), the view of the adversary corresponds to Game_3. Otherwise (i.e., $\mathbf{v}' \leftarrow \mathbb{Z}_q^{m+1}$), it corresponds to Game_4. We therefore conclude that assuming the hardness of the $\mathsf{LWE}_{n,m+1,q,\mathbb{Z}_{\alpha q}}$ problem we have

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| = \mathsf{negl}(\lambda).$$

Game_5 : In this game, we further change the way the challenge ciphertext is created when $\mathsf{coin} = 0$. If $\mathsf{coin} = 0$, the challenger samples $w \leftarrow \mathbb{Z}_q, \mathbf{w} \leftarrow \mathbb{Z}_q^m$, $z \leftarrow D_{\mathbb{Z},\alpha q}$, $\mathbf{z}' \leftarrow D_{\mathbb{Z}^{(D\ell L+2)m},\alpha' q}$, sets $\mathbf{R}^*$ as in the previous games and computes $v = w + z \in \mathbb{Z}_q$. Then, it computes

$$\mathbf{c}'^{\top} = \mathbf{w}^{\top}[\mathbf{I}_m | \mathbf{R}^*] + \mathbf{z}'^{\top} \in \mathbb{Z}_q^{(D\ell L+2)m},$$

and parses $\mathbf{c}'$ appropriately into $D\ell L + 2$ size m vectors $(\mathbf{c}, \mathbf{c}_0, (\mathbf{c}_{i,j,w})_{(i,j,w)})$. Finally, it sets the challenge ciphertext as in Eq. (23). Using the same argument we made to move from Game_2 and Game_3 concerning the noise rerandomization lemma, we can check that the above change alters the distribution of the challenge ciphertext only negligibly. Thus, we have

$$|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| = \mathsf{negl}(\lambda).$$

Game_6 : In this game, we change the challenge ciphertext to be a random vector, regardless of the value of coin. Namely, the challenger creates the challenge ciphertext $(c, \mathbf{c}, \mathbf{c}_0, (\mathbf{c}_{i,j,w})_{(i,j,w)}) \in \mathbb{Z}_q \times \mathbb{Z}_q^{(D\ell L+2)m}$ by properly formatting $(D\ell L + 2)m + 1$ random elements from $\mathbb{Z}_q$. In this game, the value coin is independent from the view of $\mathcal{A}$. Therefore, $\Pr[\mathsf{E}_6] = 1/2$.

It remains to upper bound $|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]|$. Since Game_5 and Game_6 differs only in the creation of the challenge ciphertext when $\mathsf{coin} = 0$, we focus on this case. First, it is easy to see that c is uniformly random over $\mathbb{Z}_q$ and independent of the other terms of the ciphertext in both games. Therefore, we are left to show that the distribution of $\bar{\mathbf{c}} = (\mathbf{c}, \mathbf{c}_0, (\mathbf{c}_{i,j,w})_{(i,j,w)})$ in Game_5 is negligibly close to the uniform distribution over $\mathbb{Z}_q^{(D\ell L+2)m}$. First, observe that the following distributions are negligibly close:

$$(\mathbf{A}, \mathbf{A}\mathbf{R}^*, \mathbf{w}^{\top}, \mathbf{w}^{\top}\mathbf{R}^*) \approx (\mathbf{A}, \mathbf{A}', \mathbf{w}^{\top}, \mathbf{w}'^{\top}) \approx (\mathbf{A}, \mathbf{A}\mathbf{R}^*, \mathbf{w}^{\top}, \mathbf{w}'^{\top})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{A}' \leftarrow \mathbb{Z}_q^{n \times (D\ell L+1)m}$, $\mathbf{R}^* \leftarrow \{-1, 1\}^{m \times (D\ell L+1)m}$, $\mathbf{w} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{w}' \leftarrow \mathbb{Z}_q^{(D\ell L+1)m}$. It can be seen that the first and the second distributions are negligibly close, by applying Lem. 3 for $[\mathbf{A}^{\top}|\mathbf{w}]^{\top} \in \mathbb{Z}_q^{(n+1) \times m}$ and $\mathbf{R}^*$. It can also be seen that the second and the third distributions are negligibly close, by applying the same lemma for $\mathbf{A}$ and $\mathbf{R}^*$. Adding a noise vector $\mathbf{z}'$ to the above $\mathbf{w}^{\top}\mathbf{R}^*$ does not change the statistical distance between the distributions. Therefore, we may conclude that

$$|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]| = \mathsf{negl}(\lambda).$$

Analysis. Combining everything together, we have

$$\begin{aligned}\epsilon = \left|\Pr[\mathsf{E}_0] - \frac{1}{2}\right| &= \left|\sum_{i=0}^{5}(\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]) + \Pr[\mathsf{E}_6] - \frac{1}{2}\right| \\ &\leq \sum_{i=0}^{5}|\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]| + \left|\Pr[\mathsf{E}_6] - \frac{1}{2}\right| \\ &\leq \mathsf{negl}(\lambda).\end{aligned}$$

Therefore, the probability that $\mathcal{A}$ wins Game_0 is negligible. □

Remark 4. *As noted in Remark 2 and Appendix B.2, in some cases we can compress the size of the ciphertext by taking advantage of the underlying compatible domains $(\mathcal{X}, \mathcal{Y})$. For example, in case we construct a predicate encryption scheme for the subset conjunction predicate, we can decrease the size of the ciphertext by a factor of D.*

Acknowledgement. We would like to thank the anonymous reviewers of Asiacrypt 2016 for insightful comments. In particular, we are grateful for Takahiro Matsuda and Shota Yamada for precious comments on the earlier version of this work. We also thank Atsushi Takayasu, Jacob Schuldt and Nuttapong Attrapadung for helpful comments on the draft. The research was partially supported by JST CREST Grant Number JPMJCR1302 and JSPS KAKENHI Grant Number 17J05603.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h) ibe in the standard model. In *EUROCRYPT*, pages 553–572. Springer, 2010. 3, 13, 14

[ACF09] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In *EUROCRYPT*, pages 554–571. Springer, 2009. 9

[ACF14] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions: Relations to identity-based key encapsulation and new constructions. *Journal of Cryptology*, 27(3):544–593, 2014. 2, 4, 8

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, pages 21–40. Springer, 2011. 3, 5, 9, 10, 12, 32

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptograpcrhy in NC^0. In *FOCS*, pages 166–175, 2004. 5

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *EUROCRYPT*, pages 557–577. Springer, 2014. 6

[Att16] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In *ASIACRYPT*, pages 591–623. Springer, 2016. 6

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT*, pages 223–238. Springer, 2004. 1

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459. Springer, 2004. 4, 6, 22

[BGG+14] 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. *EUROCRYPT*, pages 533–556, 2014. 3, 5, 9, 10, 31, 32

[BGJS17] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. A note on vrfs from verifiable functional encryption. Cryptology ePrint Archive, Report 2017/051, 2017. https://eprint.iacr.org/2017/051.pdf. 2, 3

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. Cryptology ePrint Archive, Report 2017/18, 2017. https://eprint.iacr.org/2017/018.pdf. 2, 3

[BKOS00] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*. Springer, 2000. 44

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013. 13

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *CCS*, pages 131–140. ACM, 2010. 2, 4, 8

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters ibe scheme. In *EUROCRYPT*, pages 407–424. Springer, 2009. 4, 15

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554. Springer, 2007. 1, 2, 3, 12, 21, 44

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system abe in prime-order groups via predicate encodings. In *EUROCRYPT*, pages 595–624. Springer, 2015. 5

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. Springer, 2010. 1, 3, 4, 6, 14, 22

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *PKC*, pages 416–431. Springer, 2005. 4, 8

[FHPS13] Eduarda SV Freire, Dennis Hofheinz, Kenneth G Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In *CRYPTO*, pages 513–530. Springer, 2013. 22

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. Cryptology ePrint Archive, Report 2017/021, 2017. https://eprint.iacr.org/2017/021.pdf. 2, 3

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. Cryptology ePrint Archive, Report 2017/274, to appear in FOCS 2017. https://eprint.iacr.org/2017/274.pdf. 3, 5

[GMW15] Romain Gay, Pierrick Méaux, and Hoeteck Wee. Predicate encryption for multi-dimensional range queries from lattices. In *PKC*, pages 752–776. Springer, 2015. 3, 5, 18, 21, 44

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. 4

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, pages 89–98. ACM, 2006. 1

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008. 14

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, pages 550–574. Springer, 2015. 5, 7, 46, 47, 49

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554. ACM, 2013. 3

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *CRYPTO*, pages 503–523. Springer, 2015. 3, 5, 44

[HJ16] Dennis Hofheinz and Tibor Jager. Verifiable random functions from standard assumptions. In *TCC*, pages 336–362. Springer, 2016. 2, 4, 14

[HK08] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In *CRYPTO*, pages 21–38. Springer, 2008. 1, 22

[HW10] Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In *EUROCRYPT*, pages 656–672. Springer, 2010. 2, 4, 9

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: a new representation with applications to round-efficient secure computation. In *FOCS*, 2000. 5

[Jag15] Tibor Jager. Verifiable random functions from weaker assumptions. In *TCC*, pages 121–143. Springer, 2015. 2, 4, 8, 15, 22

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *EUROCRYPT*, pages 146–162, 2008. 1, 2, 12

[KY16] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: more compact ibes from ideal lattices and bilinear maps. In *ASIACRYPT*, pages 682–712. Springer, 2016. 10, 14, 15, 16

[LPRTJ05] Alexander E Litvak, Alain Pajor, Mark Rudelson, and Nicole Tomczak-Jaegermann. Smallest singular value of random matrices and geometry of random polytopes. *Advances in Mathematics*, 195(2):491–523, 2005. 13

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the dh-ddh separation. In *CRYPTO*, pages 597–612. Springer, 2002. 9

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. Springer, 2012. 10, 14, 33, 43, 48

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *FOCS*, pages 120–130. IEEE, 1999. 2, 11

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. ACM, 2009. 13

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM Press, 2005. 13

[SBC+07] Elaine Shi, John Bethencourt, TH Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *S&P*, pages 350–364. IEEE, 2007. 1, 2, 3, 21, 44

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473. Springer, 2005. 1

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010. 16

[Ver11] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. Lecture Notes, 2011. Available at http://www-personal.umich.edu/romanv/papers/non-asymptotic-rmt-plain.pdf. 43

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In *TCC*, pages 616–637. Springer, 2014. 3, 5

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. Cryptology ePrint Archive, Report 2017/276, to appear in FOCS 2017. http://eprint.iacr.org/2017/276. 3, 5

[Yam16] Shota Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In *EUROCRYPT*, pages 32–62. Springer, 2016. 15

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice ibes and verifiable random functions via generalized partitioning techniques. Cryptology ePrint Archive, Report 2017/096, to appear in CRYPTO 2017. http://eprint.iacr.org/2017/096. 2, 4, 5, 6, 7, 8, 9, 16, 22, 29, 46, 47, 48, 50

[ZCZ16] Jian Zhang, Yu Chen, and Zhenfeng Zhang. Programmable hash functions from latties: Short signatures and ibes with small key sizes. In *CRYPT0*, pages 303–332. Springer, 2016. 1, 22

A Proof of Lemma 2

Proof. We first show $\mathbf{RU}$ is subgaussian with parameter $B\sqrt{mk}$. Note that we say that a random matrix $\mathbf{X}$ is *subgaussian* with parameter $\sigma > 0$ if all of its one-dimensional marginals $\mathbf{u}^\top \mathbf{X}\mathbf{v}$ for unit vectors $\mathbf{u}, \mathbf{v}$ are subgaussian with parameters σ, i.e., $\mathbb{E}[\exp(s \cdot \mathbf{u}^\top \mathbf{X}\mathbf{v})] \leq \exp(\sigma^2 s^2/2)$. Observe that

$$\mathbf{u}^\top \mathbf{R}\mathbf{U}\mathbf{v} = \sum_{i=1}^{\ell}\sum_{t=1}^{m} \mathbf{R}_{i,t}\Big(u_i \sum_{j=1}^{m} \mathbf{U}_{t,j} v_j\Big),$$

where $\mathbf{R}_{i,t}$ is the (i,t)-th element of $\mathbf{R}$. (Other terms $\mathbf{U}_{t,j}, \mathbf{u}_i, \mathbf{v}_j$ are defined analogously.) Then, we have

$$\begin{aligned}
\mathbb{E}[\exp(s \cdot \mathbf{u}^\top \mathbf{R}\mathbf{U}\mathbf{v})] &= \mathbb{E}\Big[\exp\Big(\sum_{i=1}^{\ell}\sum_{t=1}^{m} \mathbf{R}_{i,t}\big(u_i \sum_{j=1}^{m} \mathbf{U}_{t,j} v_j\big)\Big)\Big] \\
&= \prod_{i=1}^{\ell}\prod_{t=1}^{m} \mathbb{E}\Big[\exp\Big(s\mathbf{R}_{i,t}\big(u_i \sum_{j=1}^{m} \mathbf{U}_{t,j} v_j\big)\Big)\Big] \\
&\leq \prod_{i=1}^{\ell}\prod_{t=1}^{m} \exp\Big(B^2 s^2 \big(u_i \sum_{j=1}^{m} \mathbf{U}_{t,j} v_j\big)^2/2\Big) \qquad (24) \\
&\leq \prod_{i=1}^{\ell}\prod_{t=1}^{m} \exp\Big(B^2 s^2 u_i^2 \big(\sum_{j=1}^{m} \mathbf{U}_{t,j}^2\big)\big(\sum_{j=1}^{m} v_j^2\big)/2\Big) \qquad (25) \\
&\leq \exp(B^2 s^2 \sum_{i=1}^{\ell}\sum_{t=1}^{m} u_i^2 k/2) \qquad (26) \\
&= \exp(B^2 s^2 mk/2),
\end{aligned}$$

where Eq. (24) follows from the fact that any B-bounded symmetric random variable X (i.e., $|X| \leq B$) is a subgaussian with parameter B, Eq. (25) follows from the CauchySchwarz inequality, and Eq. (26) follows from the fact that $\mathbf{v}$ is a unit vector and that there are at most k ones in each row of $\mathbf{U}$. Hence, we have that $\mathbf{RU}$ is a subgaussian parameter with parameter $B\sqrt{mk}$. Finally, using the Lem. 2.9 of [MP12] (See [Ver11] for further details), we obtain the statement in the above lemma. □

B A Note on MultD-Eq Predicates

B.1 On the Expressiveness of MultD-Eq Predicates

For completeness, we discuss here how we can encode the predicates presented in Sec. 4.3 as MultD-Eq predicates with appropriate compatible domains $(\mathcal{X}, \mathcal{Y})$. This will show that MultD-Eq predicates are at least as expressive as any predicate that are used in motivating applications for

PE schemes. Despite the syntactical differences, the followings are a compilation of many previous works, e.g., [BW07, SBC+07, GVW15]. Furthermore, it should be noted that the following encoding is only one example; there are possibly many more "efficient" ways of encoding the predicates as $\mathsf{MultD\text{-}Eq}$ predicates, where the meaning of efficient may depend on the application in one's mind.

Bit-fixing predicates. Let $P^{\mathsf{BF}}_{\mathbf{v}} : \{0,1\}^\ell \to \{0,1\}$ for $\mathbf{v} \in \{0,1,?\}$. Without loss of generality, we set "?" to be 2. Then, map $\mathbf{v} \in \{0,1,2\}^\ell$ and $\mathbf{x} \in \{0,1\}^\ell$ to the following domains $\mathcal{X}_{\mathsf{BF}}, \mathcal{Y}_{\mathsf{BF}} \subseteq \mathbb{Z}_3^{\ell \times 2}$:

$$\mathbf{x} \to \bar{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_1 & 2 \\ \mathbf{x}_2 & 2 \\ \vdots & \vdots \\ \mathbf{x}_\ell & 2 \end{bmatrix} \in \mathcal{X}_{\mathsf{BF}}, \qquad \mathbf{v} \to \bar{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_1 \\ \mathbf{v}_2 & \mathbf{v}_2 \\ \vdots & \vdots \\ \mathbf{v}_\ell & \mathbf{v}_\ell \end{bmatrix} \in \mathcal{Y}_{\mathsf{BF}}.$$

It is easy to check that $\mathcal{X}_{\mathsf{BF}}, \mathcal{Y}_{\mathsf{BF}} \subseteq \mathbb{Z}_3^{\ell \times 2}$ are indeed compatible with the $\mathsf{MultD\text{-}Eq}$ predicates, i.e., for all $i \in [\ell]$ there exists at most one $j \in [2]$ such that $\bar{\mathbf{x}}_{i,j} = \bar{\mathbf{v}}_{i,j}$. Furthermore, we have $P^{\mathsf{BF}}_{\mathbf{v}}(\mathbf{x}) = 1$ if and only if $\mathsf{MultD\text{-}Eq}_{\bar{\mathbf{v}}}(\bar{\mathbf{x}}) = 1$.

Equality conjunction predicates. Let $P^{\mathsf{EC}}_{\mathbf{v}} : \Sigma^\ell \to \{0,1\}$ for $\mathbf{v} \in \Sigma^\ell$. The most natural way is to simply map Σ^ℓ into $\mathbb{Z}^\ell_{|\Sigma|}$, and define $\mathcal{X}_{\mathsf{EC}} = \mathcal{Y}_{\mathsf{EC}} = \mathbb{Z}^{\ell \times 1}_{|\Sigma|}$. This domain is trivially compatible with the $\mathsf{MultD\text{-}Eq}$ predicate, and we have $P^{\mathsf{EC}}_{\mathbf{v}}(\mathbf{x}) = 1$ if and only if $\mathsf{MultD\text{-}Eq}_{\bar{\mathbf{v}}}(\bar{\mathbf{x}}) = 1$. Similarly, we can also map Σ into $\mathbb{Z}^{w_2}_{w_1}$ where $w_1^{w_2} = |\Sigma|$, and define $\mathcal{X}_{\mathsf{EC}} = \mathcal{Y}_{\mathsf{EC}} = \mathbb{Z}^{\ell w_2 \times 1}_{w_1}$.

Subset conjunction predicates. For simplicity of presentation, we first consider an encoding for the subset predicate $P^{\mathsf{Sub}}_{\mathsf{T}} : 2^\Sigma \to \{0,1\}$ for $\mathsf{T} \in 2^\Sigma$. Further, we assume that all the inputs to $P^{\mathsf{Sub}}_{\mathsf{T}}(\cdot)$ have fixed cardinality of $n \leq |\Sigma|$ (as in the case for the subset predicate embedded in the modified admissible hash function. See. Sec. 5.1). Below, let $\mathsf{T} = \{t_1, \cdots, t_m\}$, $\mathsf{S} = \{s_1, \cdots, s_n\}$ where $m = |\mathsf{T}|$, and view elements in Σ as elements in $\mathbb{Z}_{|\Sigma|}$. Then, we can map the sets $\mathsf{T}, \mathsf{S} \in 2^\Sigma$ to matrices in the following domains $\mathcal{X}_{\mathsf{Sub}}, \mathcal{Y}_{\mathsf{Sub}} \subseteq \mathbb{Z}^{m \times n}_{|\Sigma|}$:

$$\mathsf{T} \to \bar{\mathsf{T}} = \begin{bmatrix} t_1 & t_1 & \cdots & t_1 \\ t_2 & t_2 & \cdots & t_2 \\ \vdots & \vdots & \cdots & \vdots \\ t_m & t_m & \cdots & t_m \end{bmatrix} \in \mathcal{X}_{\mathsf{Sub}}, \qquad \mathsf{S} \to \bar{\mathsf{S}} = \begin{bmatrix} s_1 & s_2 & \cdots & s_n \\ s_1 & s_2 & \cdots & s_n \\ \vdots & \vdots & \cdots & \vdots \\ s_1 & s_2 & \cdots & s_n \end{bmatrix} \in \mathcal{Y}_{\mathsf{Sub}}. \tag{27}$$

It can be checked that $\mathcal{X}_{\mathsf{Sub}}, \mathcal{Y}_{\mathsf{Sub}} \in \mathbb{Z}^{m \times n}_{|\Sigma|}$ are compatible with the $\mathsf{MultD\text{-}Eq}$ predicates. Furthermore, we have $P^{\mathsf{Sub}}_{\mathsf{T}}(\mathsf{S}) = 1$ if and only if $\mathsf{MultD\text{-}Eq}_{\bar{\mathsf{T}}}(\bar{\mathsf{S}}) = 1$.

To get rid of the restriction that every input to $P^{\mathsf{Sub}}_{\mathsf{T}}(\cdot)$ needs to have cardinality n, we can use the embedding given in [GMW15], Sec. 3.2, where they map T, S to domains in $\mathbb{Z}_3^{m \times |\Sigma|}$. Finally, to obtain an encoding for the subset conjunction predicate we can simply concatenate the encodings of the subset predicates for each T_i.

Range conjunction predicates. We can use the tree data structure for storing intervals known as *segment trees* to encode range conjunction predicates as $\mathsf{MultD\text{-}Eq}$ predicates. Since the encoding is classical and rather contrived, we only present the results here, and refer the readers to [BKOS00], Chap.10 and [GMW15], Sec. 3.3 for further details. In particular, we can encode range conjunction predicates $P^{\mathsf{RC}}_{[\mathbf{a};\mathbf{b}]} : [T]^\ell \to \{0,1\}$ as $\mathsf{MultD\text{-}Eq}$ predicates with compatible domains $\mathcal{X}_{\mathsf{RC}}, \mathcal{Y}_{\mathsf{RC}}$ in $\mathbb{Z}^{\ell \times \lceil \log T \rceil}_{T+1}$.

B.2 Exploitable Structures for More Efficient PES Schemes

Here we comment on Remark. 2. In some cases, the compatible domains $\mathcal{X}, \mathcal{Y}$ for the multidimensional predicates MultD-Eq may have additional structures that we can exploit to obtain a more efficient predicate encoding PES scheme. We illustrate this in the following using as example the subset predicate implicit in our VRF construction from Sec. 5.2.

For our VRF construction, we used a special type of subset predicate $P^{\mathsf{Sub}}_{\bar{\mathsf{T}}} : 2^{\Sigma} \to \{0,1\}$ where the inputs have fixed cardinality of n, as discussed in Appendix B.1. We showed in Eq. (27) that this particular subset predicate can be encoded as a MultD-Eq predicate with compatible domains $(\mathcal{X}_{\mathsf{Sub}}, \mathcal{Y}_{\mathsf{Sub}}) \in \mathbb{Z}_{|\Sigma|}^{m \times n} \times \mathbb{Z}_{|\Sigma|}^{m \times n}$. Therefore, for example, by Lem. 8 we can construct a functionality preserving predicate encoding scheme $\mathsf{PES}_{\mathsf{FP}}$ for the the subset predicate with $p = |\Sigma|, D = m, \ell = n$. Namely, for any $\bar{\mathsf{S}} \in \mathcal{X}_{\mathsf{Sub}}$ and $\bar{\mathsf{T}} \in \mathcal{Y}_{\mathsf{Sub}}$ we have

$$\begin{aligned}
\mathsf{EncInpt}_{\mathsf{FP}}(\bar{\mathsf{S}}) &\to \hat{\bar{\mathsf{S}}} = \left(\bar{\mathsf{S}}_{i,j,k}\right)_{(i,j,k)\in[m]\times[n]\times[\zeta]} \\
\mathsf{EncPred}_{\mathsf{FP}}(\mathsf{MultD\text{-}Eq}_{\bar{\mathsf{T}}}) &\to \hat{C}^{\mathsf{Sub}}_{\bar{\mathsf{T}}}, \\
&\text{where} \quad \hat{C}^{\mathsf{Sub}}_{\bar{\mathsf{T}}}(\hat{\bar{\mathsf{S}}}) = \prod_{i=1}^{m}\sum_{j=1}^{n}\prod_{k=1}^{\zeta}\left((1-\hat{\bar{\mathsf{T}}}_{i,j,k}) + (-1+2\hat{\bar{\mathsf{T}}}_{i,j,k})\cdot\hat{\bar{\mathsf{S}}}_{i,j,k}\right),
\end{aligned}$$

where $\zeta = \lfloor \log(|\Sigma|) \rfloor + 1$ and $\bar{\mathsf{T}}_{i,j,k}, \bar{\mathsf{S}}_{i,j,k}$ are the k-th bit of the binary representation of $\bar{\mathsf{T}}_{i,j}, \bar{\mathsf{S}}_{i,j}$, respectively. However, as it can be observed from Eq. (27), for all $k \in [\zeta]$, we have $\bar{\mathsf{T}}_{i,1,k} = \bar{\mathsf{T}}_{i,j,k}$ for all $j \in [m]$, and $\bar{\mathsf{S}}_{1,j,k} = \bar{\mathsf{S}}_{i,j,k}$ for all $i \in [n]$. Therefore, we can in fact consider a more efficient predicate encoding scheme $\mathsf{PES}'_{\mathsf{FP}}$ that takes advantage of this redundancy:

$$\begin{aligned}
\mathsf{EncInpt}'_{\mathsf{FP}}(\bar{\mathsf{S}}) &\to \hat{\bar{\mathsf{S}}} = \left(\bar{\mathsf{S}}_{1,j,k}\right)_{(j,k)\in[n]\times[\zeta]} \\
\mathsf{EncPred}'_{\mathsf{FP}}(\mathsf{MultD\text{-}Eq}_{\bar{\mathsf{T}}}) &\to \hat{C}^{\mathsf{Sub}}_{\bar{\mathsf{T}}}, \\
&\text{where} \quad \hat{C}^{\mathsf{Sub}}_{\bar{\mathsf{T}}}(\hat{\bar{\mathsf{S}}}) = \prod_{i=1}^{m}\sum_{j=1}^{n}\prod_{k=1}^{\zeta}\left((1-\hat{\bar{\mathsf{T}}}_{1,j,k}) + (-1+2\hat{\bar{\mathsf{T}}}_{1,j,k})\cdot\hat{\bar{\mathsf{S}}}_{i,1,k}\right).
\end{aligned}$$

Note that this encoding scheme (written slightly differently using the symmetry of $\hat{\bar{\mathsf{T}}}_{1,j,k}$ and $\hat{\bar{\mathsf{S}}}_{i,1,k}$) is what we present in Sec. 5.2, Eq. (13). This extra optimization allows us to decrease a factor of m in the output size of $\mathsf{EncInpt}_{\mathsf{FP}}$. Since this idea translates to $\mathsf{PES}_{\mathsf{Lin}}$ as well, in applications such as the predicate encoding scheme we provide in Sec. 6.2, this will directly yield a PE scheme with shorter ciphertexts by a factor of D.

C Omitted Proofs for Our VRF in Sec. 5.4

Here, we show correctness, unique provability and pseudorandomness of our VRF. The proofs follow closely to the ones given in Sec. 5.3. We omit the proof for the unique provability, since it is the same as the one given in Sec. 5.3.

Theorem 5 (Correctness)**.** *Our VRF from Sec. 5.4 forms a correct verifiable random function.*

Proof. We first prove the correctness of the scheme. It is easily seen that when Gen and Eval are properly run, then it passes Step 1, 2, 3 of the verification algorithm. Next, observe that for all $i \in [\eta]$ we have

$$\prod_{S\subseteq[\zeta]} \Phi_{i,S}^{\alpha_S} = \prod_{S\subseteq[\zeta]} \left(e(g^{\prod_{k\in L_S} w_{i,k}}, g^{\prod_{k\in R_S} w_{i,k}})\right)^{\alpha_S}$$

$$
\begin{aligned}
&= \prod_{S\subseteq[\zeta]} \left(e(g,g)^{\prod_{k\in S} w_{i,k}}\right)^{\alpha_S} \\
&= e(g,g)^{\sum_{S\subseteq[\zeta]} \alpha_S \prod_{k\in S} w_{i,k}} \\
&= e(g,g)^{p(w_{i,1},\cdots,w_{i,\zeta})}
\end{aligned}
$$

Since $\theta_i = p(w_{i,1}, \cdots, w_{i,\zeta})$, the first equation in Step 4 holds. The equality of the rest of the equations in Step 4 follow using the additional observation that $\theta_{[1:i']} \cdot \theta_{i'+1} = \theta_{[1:i'+1]}$ for $i' \in [\eta-1]$, where $\theta_{[1:1]} = \theta_1$. Finally, since by definition $\pi_{[1:\eta]} = g^{\theta_{[1:\eta]}} = g^{\theta}$, Step 5 holds. This completes the proof of the correctness of the scheme. □

The proof of pseudorandomness follows very closely to the proof given in Sec. 5.3. Notably, the VRF is proven under the L-DDH assumption where $L = \eta\zeta = \omega(\log^2 \lambda)$. Therefore, to avoid being redundant, we point out the main differences between the proof in Sec. 5.3 and restrict ourselves to an overview of the security proof.

Proof Sketch. At a high level, the strategy of the proof is the same; we show that we can simulate all the components in the verification key and a valid output Y^* for the challenge input X^* using the L-DDH instance $\{g^{\alpha^i}\}_{i\in[\eta\zeta]}$. Here, note that if we can simulate a valid output Y^*, we can also simulate a valid proof for any input X such that $\mathsf{T} \not\subseteq \mathsf{S}(X)$. We first show that the challenger can correctly simulate the verification key. As in Game_2, Eq. (15) of the previous proof, the challenger sets

$$
w_0 = \tilde{w}_0 \cdot \alpha, \quad w_{i,k} = \tilde{w}_{i,k} \cdot \alpha + t_{i,k} \quad \text{for} \quad (i,k) \in [\eta] \times [\zeta].
$$

To create the rest of $(g^{\mathcal{P}(L_i)}, g^{\mathcal{P}(R_i)})_{i\in[\eta]}$, it can simply use $\{g^{\alpha^i}\}_{i\in[\eta\zeta]}$ since the terms in $\mathcal{P}(L_i), \mathcal{P}(R_i)$ are at most degree ζ. Recall $L_i = (w_{i,k})_{k\in[\lfloor\zeta/2\rfloor]}$ and $R_i = (w_{i,k})_{k\in[\lfloor\zeta/2\rfloor+1:\zeta]}$. Furthermore, since we use the same degree $\eta\zeta$ polynomial $\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z})$ as in Eq. (16) to embed the partitioning function $\mathsf{F}_{\mathsf{MAH}}$, we can correctly simulate the proof as in Game_5 using $\{g^{\alpha^i}\}_{i\in[\eta\zeta]}$. Thus, we have that our VRF is adaptively pesudorandom. □

Combining everything together, our second VRF satisfies all the desired properties under the L-DDH assumption where $L = \omega(\log^2 \lambda)$.

D Other Applications: Improvement on [Yam17] IBE

In this section, we give an (informal) overview on how to make the identity-based encryption (IBE) scheme of [Yam17] more efficient using the preicate encoding scheme of Eq. (3) in Sec. 2. Notably, we are able to lower the approximation factor of the LWE problem from $\tilde{O}(n^{11})$ to $\tilde{O}(n^{5.5})$ by exploiting the additive structure of our embedded polynomial and with some additional techniques concerning random matrices used in our proof of Lem. 2. Furthermore, we are able to parallelize the encryption and key generation algorithm, whereas the algorithms of [Yam17] are inherently unparallelizable since they rely heavily on the sequential matrix multiplication technique of [GV15].

Recall that [Yam17] provides a modular construction of IBEs. They first define the notion of *compatible algorithms* for partitioning functions (See Def. 10). Then, they propose a generic construction of IBE schemes from a partitioning function with its associating compatible algorithms. In particular, they obtain an IBE scheme by instantiating this framework with the compatible algorithms for the modified admissible hash function $\mathsf{F}_{\mathsf{MAH}}$ (See Def. 11). Below, we provide the definition of compatible algorithms.

Definition 13. ([Yam17], Def. 8) *We say that the deterministic algorithms* $(\mathsf{Encode}, \mathsf{PubEval}, \mathsf{TrapEval})$ *are* δ*-compatible with a function family* $\{\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \{0,1\}\}$ *if they are efficient and satisfy the following properties:*

- $\mathsf{Encode}(K \in \mathcal{K}) \to \kappa \in \{0,1\}^u$
- $\mathsf{PubEval}(X \in \mathcal{X}, \{\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}\}_{i \in [u]}) \to \mathbf{B}_X \in \mathbb{Z}_q^{n \times m}$
- $\mathsf{TrapEval}(K \in \mathcal{K}, X \in \mathcal{X}, \mathbf{A} \in \mathbb{Z}_q^{n \times m}, \{\mathbf{R}_i \in \mathbb{Z}^{m \times m}\}_{i \in [u]}) \to \mathbf{R}_X \in \mathbb{Z}^{m \times m}$

We require the following to hold:

$$\mathsf{PubEval}(K, X, \mathbf{A}, \{\mathbf{A}\mathbf{R}_i + \kappa_i \mathbf{G}\}_{i \in [u]}) = \mathbf{A}\mathbf{R}_X + \mathsf{F}(K, X)\mathbf{G},$$

where $\kappa_i \in \{0,1\}$ *is the* i*-th bit of* $\kappa = \mathsf{Encode}(K) \in \{0,1\}^u$*. Furthermore, if* $\mathbf{R}_i \in \{-1, 0, 1\}^{m \times m}$ *for all* $i \in [u]$*, we have* $\|\mathbf{R}_X\|_\infty \leq \delta$*.*

At a high level, $\mathsf{PubEval}$ is a public algorithm used to compute the hash of an $\mathsf{ID} \in \mathcal{X}$ and $\mathsf{TrapEval}$ is a secret algorithm used by the simulator to recover the $\mathbf{G}$-trapdoor $\mathbf{R}_X$. Therefore, since $\mathbf{R}_X$ is used as a trapdoor to sample a secret key for user X, the quality of $\mathbf{R}_X$ has a direct effect on the efficiency and required hardness assumption for LWE. In particular, the value of δ has a *quadratic* effect on the approximation factor of the LWE problem used in the underlying IBE scheme. Thus, compatible algorithms for $\mathsf{F}_{\mathsf{MAH}}$ with a smaller δ will directly yield a more efficient IBE construction.

In thier work, they (basically) used Eq. (1) to compute $\mathsf{F}_{\mathsf{MAH}}$ (See Eq. (12)) and obtained δ-compatible algorithms $(\mathsf{Encode}_{\mathsf{Yam}}, \mathsf{PubEval}_{\mathsf{Yam}}, \mathsf{TrapEval}_{\mathsf{Yam}})$ for $\mathsf{F}_{\mathsf{MAH}}$ where $\delta = \tilde{O}(\lambda^4)$, which can be obtained by plugging in the values from Thm. 1. Furthermore, due to the multiplicative structure of Eq. (1), they heavily rely on the sequential matrix multiplication technique of [GV15] in order to control the growth of δ. This is the reason why their scheme is inherently unparallelizable.

We provide two ideas to improve their scheme. First, we can do much better by using Eq. (3) to compute $\mathsf{F}_{\mathsf{MAH}}$. Namely, we use the following polynomial defined over $\mathbb{Z}_q$, which is a slight modification of Eq. (3):

$$\eta - \sum_{i=1}^{\eta} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot t_{i,k} \Big) = \begin{cases} 0 & \text{if} \quad \mathsf{T} \subseteq \mathsf{S}(X) \\ \in \{1, \cdots \eta\} & \text{if} \quad \mathsf{T} \not\subseteq \mathsf{S}(\mathsf{X}) \end{cases}, \tag{28}$$

A subtle point here is that we have to alter Def. 13 so that the function family can take output over $\mathbb{Z}_q$. Recall that in the above we required $\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) \in \{0,1\}$. However, we can easily show that for the security proof for the IBE scheme to follow through, we do not necessarily need the output to be in $\{0,1\}$, as long as we have $\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = 0$ iff $\mathsf{T} \subseteq \mathsf{S}(X)$.

For completeness, we provide the algorithms for $\mathsf{PubEval}, \mathsf{TrapEval}$ following the notations of [Yam17], Sec. 5.1. The Encode algorithm is defined as in [Yam17].

$\mathsf{PubEval}(X, \{\mathbf{B}_{i,k}\}_{(i,k) \in [\eta] \times [\zeta]} \in \mathbb{Z}_q^{n \times m})$: It first computes $S(X) = \{s_1, \cdots, s_\ell\} \subset [2\ell]$. Let $s_{j,k} \in \{0,1\}$ be the k-th bit of the binary representation of s_j. It then proceeds as follows:

1. For $(i,j,k) \in [\eta] \times [\ell] \times [\zeta]$, it sets $\mathbf{V}_{i,j,k} = (1 - s_{j,k}) \cdot \mathbf{G} + (-1 + 2s_{j,k}) \cdot \mathbf{B}_{i,k}$
2. For $(i,j) \in [\eta] \times [\ell]$, set $\mathbf{V}_{i,j,[1:1]} := \mathbf{V}_{i,j,1}$ and compute $\mathbf{V}_{i,j,[1:k+1]} := \mathbf{V}_{i,j,k+1} \cdot \mathbf{G}^{-1}(\mathbf{V}_{i,j,[1:k]})$ for $k \in [\zeta - 1]$. Then set $\mathbf{V}_{i,j} = \mathbf{V}_{i,j,[1:\zeta]}$.

3. Finally, it outputs $\mathbf{B}_X = \eta \cdot \mathbf{G} - \sum_{i\in[\eta]} \sum_{j\in[\ell]} \mathbf{V}_{i,j}$

$\mathsf{TrapEval}(\mathbf{T}, X, \mathbf{A}, \{\mathbf{R}_{i,k}\}_{(i,k)\in[\eta]\times[\zeta]} \in \mathbb{Z}_q^{n\times m})$: It first computes $S(X) = \{s_1, \cdots, s_\ell\} \subset [2\ell]$ and parses $\mathbf{T} \to (t_1, \cdots, t_{\eta'}) \subset [2\ell]$, where $\eta' < \eta$. It then sets $t_{\eta'+1} = \cdots = t_\eta = 0$. In the following, let $t_{i,k}, s_{j,k} \in \{0,1\}$ be the k-th bit of the binary representation of t_i, s_j, respectively. It then proceeds as follows:

1. For $(i,j,k) \in [\eta] \times [\ell] \times [\zeta]$, it sets $\mathbf{S}_{i,j,k} = (-1 + 2s_{j,k}) \cdot \mathbf{R}_{i,k}$
2. For $(i,j) \in [\eta]\times[\ell]$, set $\mathbf{S}_{i,j,[1:1]} := \mathbf{S}_{i,j,1}$ and compute $\mathbf{S}_{i,j,[1:k+1]} := \mathbf{R}_{i,k+1} \cdot \mathbf{G}^{-1}(\mathbf{V}_{i,j,[1:k]}) + (-1 + 2s_{j,k+1}) \cdot \mathbf{S}_{i,j,[1:k]}$ for $k \in [\zeta - 1]$, where $\mathbf{V}_{i,j,[1:k]}$ is defined as above. Then set $\mathbf{S}_{i,j} = \mathbf{S}_{i,j,[1:\zeta]}$.
3. Finally, it outputs $\mathbf{R}_X = \sum_{i\in[\eta]} \sum_{j\in[\ell]} \mathbf{S}_{i,j}$

Lemma 14. *The above algorithms* $(\mathsf{Encode}, \mathsf{PubEval}, \mathsf{TrapEval})$ *are* $m\zeta\eta\ell$*-compatible algorithms for* $\mathsf{F}_{\mathsf{MAH}}$*. In particular, if we instantiate* $\mathsf{F}_{\mathsf{MAH}}$ *using Def. 11,* $m\zeta\eta\ell = \tilde{O}(\lambda^2)$*.*

Proof. First observe the inequality

$$\begin{aligned}\|\mathbf{S}_{i,j,[k+1]}\|_\infty &= \|\mathbf{R}_{i,k+1} \cdot \mathbf{G}^{-1}(\mathbf{V}_{i,j,[1:k]}) + (-1 + 2s_{j,k+1}) \cdot \mathbf{S}_{i,j,[1:k]}\|_\infty \\ &\le m \cdot \|\mathbf{R}_{i,k+1}\|_\infty + \|\mathbf{S}_{i,j,[1:k]}\|_\infty \\ &\le m + \|\mathbf{S}_{i,j,[1:k]}\|_\infty,\end{aligned}$$

where we use $(-1 + 2s_{j,k+1}) \in \{0,1\}, \mathbf{R}_{i,k+1} \in \{-1,0,1\}^{m\times m}$. Therefore by induction, we have $\|\mathbf{S}_{i,j}\|_\infty \le m\zeta$. Hence, we obtain the bound. □

Note that the poly-log factor hidden in the $\tilde{O}(\cdot)$ notation is the same as [Yam17]. Our δ is a $O(\lambda^2)$ factor smaller than the scheme of [Yam17], and since δ has a quadratic effect on the approximation factor of the LWE problem, we are able to lower the approximation factor down by $O(\lambda^4)$. Finally, we make the following subtle observations:

- Using subgaussian arguments, the error term can be bounded by $O(\alpha'\sigma\sqrt{mq})$ instead of $O(\alpha'\sigma mq)$. (See [Yam17], Lem. 13).
- Since $\mathbf{R}_X \in \mathbb{Z}^{m\times m}$ is subgaussian with parameter δ (which follows from $\|\mathbf{R}\|_\infty \le \delta$), we have $s_1(\mathbf{R}_X) \le C \cdot \sqrt{m}\delta$ with overwhelming probability for some positive constant C ([MP12], Lem. 2.9). Therefore, we can use $\alpha' > O(\alpha\sqrt{m}\delta)$, instead of $\alpha' > O(\alpha m\delta)$. (See [Yam17], Sec. 6.2).
- Use the sampling algorithm of [MP12] to obtain $\sigma > \tilde{\Omega}(\sqrt{m}\delta)$ instead of $\sigma > \tilde{\Omega}(m\delta)$ (See. [Yam17], Lem. 3).

Combining this together, we obtain a candidate parameter selection as follows:

$$\begin{aligned} &m = O(n \log q), && q = n^2 \cdot \delta^2 \cdot \omega(\log^2 n), && \sigma = m \cdot \delta \cdot \omega(\sqrt{\log m}), \\ &\alpha q = 3\sqrt{n} && \alpha' q = 5\sqrt{n} \cdot m \cdot \delta. \end{aligned}$$

Plugging in our δ-compatible algorithm for the $\mathsf{F}_{\mathsf{MAH}}$ function, we obtain an approximation factor of $\tilde{O}(n^{5.5})$ for the LWE problem. Recall that the approximation factor of [Yam17] was $\tilde{O}(n^{11})$. Finally, we are also able to improve significantly on the parallel complexity of the IBE scheme. Notably, our compatible algorithms $(\mathsf{Encode}, \mathsf{PubEval}, \mathsf{TrapEval})$ for the modified admissible hash

function $\mathsf{F}_{\mathsf{MAH}}$ allows for high parallelization of the encryption and key generation algorithm. (Recall $\mathsf{PubEval}$ is used to compute the hash of an $\mathsf{ID} \in \mathcal{X}$.) We obtain parallel speed up because our encoded polynomial of $\mathsf{F}_{\mathsf{MAH}}$ has an additive structure, and we do not have to rely on the sequential matrix multiplication technique of [GV15] to control the growth of $\mathbf{R}_X$.

Contents