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

* The University of Tokyo, National Institute of Advanced Industrial Science and Technology (AIST). E-mail: shuichi_katsumata@it.k.u-tokyo.ac.jp

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.

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 handcraft 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.

[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.

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 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].

[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].

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 $\mathsf{MultD\text{-}Eq}$ predicates. For the time being, it suffices to think of the $\mathsf{MultD\text{-}Eq}$ 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 $\mathsf{MultD\text{-}Eq}$ predicates. Then, based on these encoded $\mathsf{MultD\text{-}Eq}$ predicates, we construct our VRFs, and PE schemes for the class of $\mathsf{MultD\text{-}Eq}$ predicates. The following is a summary of our two results.

VRF. We propose two VRFs with all the desired properties. The detailed comparison between the recent efficient VRF constructions are given in Table 1. Note that we exclude 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.

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

Schemes	$\lvert\mathsf{vk}\rvert$ $\#\,\mathbb{G}$	$\lvert\pi\rvert$ $\#\,\mathbb{G}$	Assumption	Reduction Cost
[Jag15]	$O(\lambda)$	$O(\lambda)$	$O(\log(Q/\epsilon))$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})$
[HJ16]	$O(\lambda)$	$O(\lambda)$	DLIN	$O(\epsilon^{\nu+1}/\lambda Q^{\nu})$
[Yam17]: Sec. 7.1.	$\omega(\lambda \log \lambda)$	$\omega(\log \lambda)$	$\omega(\lambda \log \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})$
[Yam17]: Sec. 7.3.	$\omega(\log \lambda)$	$\omega(\sqrt{\lambda} \log \lambda)$	$\omega(\lambda \log \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})$
[Yam17]: App. C.	$\omega(\log \lambda)$	$\mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$-DDH	$O(\epsilon^2/\lambda^2 Q)$
Ours: Sec. 5.1.	$\omega(\log^2 \lambda)$	$\omega(\lambda \log^2 \lambda)$	$\omega(\log^2 \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})$
Ours: Sec. 5.3.	$\omega(\sqrt{\lambda} \log \lambda)$	$\omega(\log \lambda)$	$\omega(\log^2 \lambda)$-DDH	$O(\epsilon^{\nu+1}/Q^{\nu})$

To measure the verification key size $\lvert\mathsf{vk}\rvert$ and proof size $\lvert\pi\rvert$, we count the number of group elements in $\mathbb{G}$. Q, ϵ denotes the number of adversarial queries and advantage, respectively. ν is a constant satisfying $c = 1 - 2^{-1/\nu}$, where c is the relative distance of the underlying error correcting code $C : \{0,1\}^n \to \{0,1\}^{\ell}$.

Our constructions are inspired by the bilinear map based VRFs of [Yam17], where they noticed that an admissible hash function [BB04b, CHKP10] can be

[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.

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 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. The detailed overview and comparison are given in Table 2. We disculde the generic construction of [GVW15, GKW17, WZ17] from the table, since our primal goal was to compare the efficiency of the schemes. Our scheme achieves the best efficiency in terms of decryption time and the required modulus size q; [GMW15] requires 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. 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].

[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.

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

Schemes	$\lvert\mathsf{mpk}\rvert$ $\#\ \mathbb{Z}_q^{n\times m}$	$\lvert\mathsf{sk}\rvert$ $\#\ \mathbb{Z}^{2m}$	$\lvert\mathsf{ct}\rvert$ $\#\ \mathbb{Z}_q^{m}$	LWE param $1/\alpha$	Dec. Time $\#$ 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 $\mathbb{Z}_q^{2m}$.

† For fairness, we provided a more rigorous analysis for their parameter selections.

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 required matrix multiplications during encryption and key generation. Notably, our construction does not rely on the heavy 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. Details are provided the full version.

1.2 Related Work

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 $K \in \{0,1,\perp\}^\ell$ where the number of non-$\perp$ 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-$\perp$ 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 $\perp$ symbols and only has $O(\log \lambda)$ meaningful non-$\perp$ 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-$\perp$ 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 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

[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.

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* $\mathsf{MultD\text{-}Eq}$ 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 $\mathsf{MultD\text{-}Eq}$ predicates. First, the $\mathsf{MultD\text{-}Eq}$

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 is provided in Sec. 4.2, 4.3.

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 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:

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

[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.

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 $\mathsf{MultD\text{-}Eq}$ Predicates. Our proposed predicate encryption scheme for the $\mathsf{MultD\text{-}Eq}$ 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 $\mathsf{PES_{Lin}}$ for the $\mathsf{MultD\text{-}Eq}$ predicates. In the overview, we continue with our examples with the subset predicate for simplicity. The core idea is the same for the $\mathsf{MultD\text{-}Eq}$ predicates. Essentially, $\mathsf{PES_{Lin}}$ 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

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

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].

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 $\mathbf{R}_\mathsf{S}$). We believe this technique to be useful for obtaining tighter analysis on other lattice-based constructions.

3 Preliminaries

Notation. 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 $\{n, \cdots, m-1, m\}$.

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 Encryption

We use the standard syntax of predicate encryption (PE) schemes [BW07,KSW08, AFV11], where $P(X) = 1$ signifies the ability to decrypt. We briefly recall the security notion of PE schemes and refer the exact definition to the full version. In our paper, we define the notion of *selectively secure* and *weakly attribute hiding* using a standard game-based security formalization. The former notion requires

the challenge ciphertext to leak no information on the message, given the challenge attribute at the outset of the game. The latter notion also requires that the challenge ciphertext leaks no information on the attribute, if the adversary is only allowed to obtain secret keys that do no decrypt the challenge ciphertext.

3.3 Background on Lattices

For an integer $m > 0$, let $D_{\mathbb{Z}^m,\sigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter $\sigma > 0$. Other lattice notions are defined in the standard way.

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

Definition 2 (Learning with Errors). *For integers n, m, a prime $q > 2$, an error distribution over χ 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].

Gadget Matrix. We use the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ defined in [MP12]. 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 $\|\mathbf{T_G}\|_{\mathsf{GS}} \leq \sqrt{5}$. Further properties on $\mathbf{G}$ can be found in [MP12] or the full version.

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

Lemma 1. ([GPV08, ABB10, CHKP10, MP12]) *Let $n, m, q > 0$ be integers with $m > 2n\lceil \log q \rceil$.*

- $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathbf{T_A})$*: 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 $\|\mathbf{T_A}\|_{\mathsf{GS}} = O(\sqrt{n \log q})$.*
- $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{u}, \mathbf{T_A}, \sigma) \rightarrow \mathbf{e}$ *: 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 $D_{\Lambda_{\mathbf{u}}^\perp([\mathbf{A}|\mathbf{B}]),\sigma}$.*

3.4 Background on Bilinear Maps.

We define certified bilinear group generators following [HJ16]. We require that there is an efficient bilinear group generator algorithm GrpGen that on input 1^λ outputs a description Π 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 require GrpGen to be certified in the sense that there is an efficient algorithm GrpVfy that on input a description of the bilinear groups, outputs the validity of the description. Furthermore, we require that each group element has a unique encoding, which can be efficiently recognized. For the precise definition, we refer [HJ16]. The following is the hardness assumption we use in our scheme.

Definition 3 (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 $\mathcal{A}$.

4 Encoding Predicates with Arithmetic Circuits

In this section, we formalize the intuition outlined in the introduction on how to encode predicates as circuits. Here, we view predicates as simply a function $P : \mathcal{X} \to \{0,1\}$ over some domain $\mathcal{X}$ with image $\{0,1\}$. Furthermore, to capture the algebraic properties of arithmetic circuits, we adapt the view of treating circuits as polynomials and vice versa.

4.1 Predicate Encoding Scheme

We formalize our main tool: predicate encoding scheme.

Definition 4 (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* $\lambda \in \mathbb{N}$ *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* $\hat{\boldsymbol{x}} \leftarrow \mathsf{EncInpt}(1^\lambda, \boldsymbol{x}), \hat{C} \leftarrow \mathsf{EncPred}(1^\lambda, P)$*, and* $b \in \{0, 1\}$.

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 unless stated otherwise.

4.2 Encoding Multi-Dimensional Equality Predicates

Here, we propose two predicate encoding schemes for the *multi-dimensional equality predicate*[10] (MultD-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 MultD-Eq are defined over, and then formally define what they are.

Definition 5 (Compatible Domains for MultD-Eq). *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* $\mathsf{Y}_{i,j}$ *denote the* (i, j)*-th element of* X *and* Y *respectively.*

[10] This predicate is presented in the works of [GMW15] as the AND-OR-EQ predicate satisfying the so called "at most one" promise. The conceptual differences between their formalization and ours is that, 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. 6).

Definition 6 (MultD-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* MultD-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 $\mathsf{Y}_{i,j}$ denote 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 MultD-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 appropriately instantiating the compatible domains, we can embed many useful predicates into the MultD-Eq predicate. Further discussions are given in Sec. 4.3.

We now present two types of predicate encoding schemes for the MultD-Eq predicate. The correctness of the two schemes are provided in the full version.

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 $\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}$. This encoding scheme will be used for our VRF construction in Sec. 5.

Lemma 2. *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* 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 $\mathsf{PES}_{\mathsf{FP}} = (\mathsf{EncInpt}_{\mathsf{FP}}, \mathsf{EncPred}_{\mathsf{FP}})$ 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.
- $\mathsf{EncPred}_{\mathsf{FP}}(\mathsf{MultD\text{-}Eq}_{\mathsf{Y}}) \to \hat{C}_{\mathsf{Y}}$ *: 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* $S_0 = \{0\}$, $S_1 = \{1\} \subset \mathbb{Z}_q$.

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 $\{0,1\}$. This encoding scheme will be used for our lattice-based PE scheme for the $\mathsf{MultD\text{-}Eq}$ predicate in Sec. 6.

Lemma 3. *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* $\mathsf{MultD\text{-}Eq}$ *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* $\mathsf{PES}_{\mathsf{Lin}} = (\mathsf{EncInpt}_{\mathsf{Lin}}, \mathsf{EncPred}_{\mathsf{Lin}})$ *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* $w-1$[11] *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.*

[11] 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 $n \in \mathbb{N}$.

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

Remark 1. In some applications, the compatible domains $(\mathcal{X}, \mathcal{Y})$ for $\mathsf{MultD\text{-}Eq}$ will have some additional structures that we can exploit to obtain more efficient encoding schemes. For 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 $\mathsf{EncInpt}$ by a factor of D by discarding the redundant terms.

4.3 Expressiveness of Multi-Dimensional Equality Predicates

Here we comment on the expressiveness of the multi-dimensional equality predicates $\mathsf{MultD\text{-}Eq}$. Notably, many predicates that come up in cryptography (e.g., bit-fixing, subset conjunction, range conjunction predicates) 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 those predicates. We provide a thorough discussion in the full version.

5 Verifiable Random Functions

Modified Admissible Hash Functions In this work, we use the *modified admissible hash function* of [Yam17] to prove security of our VRF. 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 7. *(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*

$$\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X) = \begin{cases} 0, & \textit{if} \ \ \mathsf{T} \subseteq \mathsf{S}(X) \\ 1, & \textit{otherwise} \end{cases} \qquad \textit{where} \quad \mathsf{S}(X) = \{2i - C(X)_i \mid i \in [\ell]\}. \tag{8}$$

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

We also need the notion of *partitioning functions* as introduced in [Yam17] to prove security of our VRF. Informally, there exists a PPT algorithm PrtSmp

called the partitioning function that given some polynomial function $Q(\lambda)$ and a noticeable function $\epsilon_0(\lambda)$, outputs a set $\mathsf{T} \in \mathcal{K}_{\mathsf{MAH}}$ such that for all $X^*, \{X_i\}_{i=1}^{Q} \in \mathcal{X}_{\mathsf{MAH}}$ the probability of $\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^*) = 0 \wedge \bigwedge_{i=1}^{Q} \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^{(i)}) = 1$ is noticeable. The concrete definition can be found in [Yam17] or in the full version.

5.1 Construction

Below, $n, \ell, \eta, \mathsf{S}(\cdot)$ are the parameters and function specified by the modified admissible hash function and ζ is set as $\lfloor \log p \rfloor + 1$. Note that $n, \ell = \Theta(\lambda)$ and $\eta = \omega(\log \lambda)$.

$\mathsf{Gen}(1^\lambda)$: 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

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

$\mathsf{Eval}(\mathsf{sk}, X)$: 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

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

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

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

$\mathsf{Verify}(\mathsf{vk}, X, (Y, \pi))$: 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 $\left(\Pi, g, h, g_0, \left(g_{i,k}\right)_{(i,k)\in[\eta]\times[\zeta]}\right)$.
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 above form

4. It holds that for all $i' \in [\eta - 1]$ and $(i, j, k') \in [\eta] \times [\ell] \times [\zeta - 1]$,

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

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.

5.2 Correctness, Unique Provability, and Pseudorandomness

Correctness and unique provability for the above scheme can be shown by simple calculation. The proof is provided in the full version. The following theorem addresses the pseudorandomness of the scheme.

Theorem 1 (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, by the definition of partitioning functions for the admissible hash function, if we run $\mathsf{T} \leftarrow \mathsf{PrtSmp}_{\mathsf{MAH}}(1^\lambda, Q(\lambda), \epsilon_0(\lambda))$, 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 may 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 the partitioning function $\mathsf{T} \leftarrow \mathsf{PrtSmp}_{\mathsf{MAH}}(1^\lambda, Q(\lambda), \epsilon_0(\lambda))$. As noted earlier, we have $|\mathsf{T}| \subseteq [2\ell]$ and $|\mathsf{T}| < \eta$. Then, it checks whether the following condition holds:

$$\mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^{(1)}) = 1 \ \wedge \ \cdots = \ \wedge \ \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^{(Q)}) = 1 \ \wedge \ \mathsf{F}_{\mathsf{MAH}}(\mathsf{T}, X^*) = 0$$
$$\iff \ \left(\mathsf{T} \not\subseteq \mathsf{S}(X^{(1)})\right) \ \wedge \ \cdots \ \wedge \ \left(\mathsf{T} \not\subseteq \mathsf{S}(X^{(Q)})\right) \ \wedge \ \left(\mathsf{T} \subseteq \mathsf{S}(X^*)\right) \qquad (9)$$

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 (9) holds, the challenger sets $\mathsf{coin}' = \widehat{\mathsf{coin}}$. By the property of the partitioning function we have $|\Pr[\mathsf{E}_1] - 1/2| \geq \tau$ for infinitely many λ, where $\tau = \tau(\lambda)$ is a noticeable function. See the full version for a formal treatment concerning the partitioning function.

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

$$w_0 = \tilde{w}_0 \cdot \alpha, \quad w_{i,k} = \tilde{w}_{i,k} \cdot \alpha + t_{i,k} \quad \text{for} \quad (i,k) \in [\eta] \times [\zeta], \tag{10}$$

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

$$\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X)}(\mathsf{Z}) = \prod_{i=1}^{\eta} \sum_{j=1}^{\ell} \prod_{k=1}^{\zeta} \Big((1 - s_{j,k}) + (-1 + 2s_{j,k}) \cdot (\tilde{w}_{i,k}\mathsf{Z} + t_{i,k}) \Big), \tag{11}$$

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 4. *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* $\mathsf{T} \subseteq \mathsf{S}(X)$.

This can be checked by the property of the functionality preserving encoding scheme $\mathsf{PES}_{\mathsf{FP}}$ scheme. We omit the proof of this lemma to the full version. 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

the Eval algorithm:

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

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 (9) 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 (9) 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

$$\begin{aligned} Y &= e(g^{\mathsf{R}_{\mathsf{T} \subseteq \mathsf{S}(X)}(\alpha)/\tilde{w}_0}, h), \\ \pi &= \Big(\pi_0 = g^{\mathsf{R}_{\mathsf{T} \subseteq \mathsf{S}(X)}(\alpha)/\tilde{w}_0}, \\ &\qquad \big(\pi_{i'} = g^{\theta^X_{i'}(\alpha)}\big)_{i' \in [\eta]}, \big(\pi_{i,j,k'} = g^{\theta^X_{i,j,k'}(\alpha)}\big)_{(i,j,k') \in [\eta] \times [\ell] \times [\zeta]} \Big). \end{aligned}$$

Note that existence of such a polynomial $\mathsf{P}_{\mathsf{T} \subseteq \mathsf{S}(X)}(\mathsf{Z})$ is guaranteed by Lem. 4. 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. 4. It then sets,

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

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

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

$$= e(g^{\mathsf{P}_{\mathsf{T}\subseteq\mathsf{S}(X^*)}(\alpha)/w_0}, h) = e(g^{\theta/w_0}, h).$$

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 $Y^* \leftarrow \mathbb{G}_T$. We show in the full version that 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 $|\Pr[\mathsf{E}_6]-1/2| = |\Pr[\mathsf{E}_1]-1/2+\sum_{i=1}^{5}(\Pr[\mathsf{E}_{i+1}] - \Pr[\mathsf{E}_i])| \geq |\Pr[\mathsf{E}_1] - 1/2| - \sum_{i=1}^{5}|\Pr[\mathsf{E}_{i+1}] - \Pr[\mathsf{E}_i]| \geq \tau(\lambda) - \mathsf{negl}(\lambda)$, 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.

5.3 Achieving Smaller Proof Size

In this section, we propose a variant of the VRF presented in Sec. 5.1 with a much shorter proof size. In particular, using the idea outlined in the technical overview, we obtain a VRF with proof size $|\pi| = \omega(\log \lambda)$ and verification key size $|\mathsf{vk}| = \omega(\sqrt{\lambda}\log\lambda)$.

Preparation. To make the presentation more clean, we define the notion of *power tuples*. 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

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

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

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

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

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

$\mathsf{Verify}(\mathsf{vk}, X, (Y, \pi))$: 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. $\mathsf{GrpVfy}(\Pi) = \mathsf{GrpVfy}(\Pi, s) = 1$ 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 $i' \in [3, \eta]$,

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

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. 5.1. The proof is provided in the full version.

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 $\mathsf{MultD\text{-}Eq}$. This directly yields predicate encryption schemes for all the predicates presented in Sec. 4.3. Due to the symmetry of the $\mathsf{MultD\text{-}Eq}$ 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

The following definition gives a sufficient condition for constructing predicate encryption schemes. For discussions and comparisons with the related definition of [BGG+14] for attribute-based encryption schemes are given in the full version.

Definition 8. *We say 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:*

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

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.*

The linear predicate encoding scheme $\mathsf{PES}_{\mathsf{Lin}}$ for the MultD-Eq predicates (Sec. 4.2, Lem. 3) provides us with a family of arithmetic circuits $\hat{\mathcal{C}}$ that allows for $\alpha_{\hat{\mathcal{C}}}$-PE enabling algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\text{ct-priv}}, \mathsf{Eval}_{\mathrm{sim}})$. In particular we have the following lemma, which we provide the proof in the full version.

Lemma 5. *There exist* $\alpha_{\hat{\mathcal{C}}}$*-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* $C > 0$.

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 the full version. 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

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

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{MultD\text{-}Eq}_\mathsf{Y})$: 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

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

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

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

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

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

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.

Correctness and Parameter Selection. We omit the correctness of our scheme and a candidate parameter selection to the full version. We note that we can chose the modulus size as small as $q = \sqrt{m} \cdot (\sqrt{D\ell p})^{-1} \cdot \alpha_{\hat{C}}^2 \cdot \omega(\log m)$. In particular, we can base security on the polynomial LWE assumption.

Security Proof. The following theorem addresses the security of the scheme.

Theorem 2. *Given 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* $\mathsf{LWE}_{n,m+1,q,D_{\mathbb{Z},\alpha q}}$.

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.

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

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.

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.

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

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.

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

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.

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

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.

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.

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.

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.

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.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554. Springer, 2007.

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.

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.

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

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.

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.

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

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.

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

GV15. Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, pages 550–574. Springer, 2015.

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

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

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

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

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

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

Jag15. Tibor Jager. Verifiable random functions from weaker assumptions. In *TCC*, pages 121–143. Springer, 2015.

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

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.

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

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. Springer, 2012.

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

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

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

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.

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

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

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.

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.

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.