Tighter Adaptive IBEs and VRFs: Revisiting Waters' Artificial Abort

Goichiro Hanaoka[1], Shuichi Katsumata[1,2], Kei Kimura[3], Kaoru Takemure[1,2], Shota Yamada[1]

[1] AIST
hanaoka-goichiro@aist.go.jp, yamada-shota@aist.go.jp
[2]PQShield
shuichi.katsumata@pqshield.com, kaoru.takemure@pqshield.com
[3]Kyushu University
kkimura@inf.kyushu-u.ac.jp

June 3, 2025

Abstract

One of the most popular techniques to prove adaptive security of identity-based encryptions (IBE) and verifiable random functions (VRF) is the *partitioning technique*. Currently, there are only two methods to relate the adversary's advantage and runtime (ϵ, T) to those of the reduction's $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}})$ using this technique: One originates to Waters (Eurocrypt 2005) who introduced the famous *artificial abort* step to prove his IBE, achieving $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon/Q), \mathsf{T} + O(Q^2/\epsilon^2))$, where Q is the number of key queries. Bellare and Ristenpart (Eurocrypt 2009) provide an alternative analysis for the same scheme removing the artificial abort step, resulting in $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon^2/Q), \mathsf{T} + O(Q))$. Importantly, the current reductions all loose quadratically in ϵ.

In this paper, we revisit this two decade old problem and analyze proofs based on the partitioning technique through a new lens. For instance, the Waters IBE can now be proven secure with $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon^{3/2}/Q), \mathsf{T} + O(Q))$, breaking the quadratic dependence on ϵ. At the core of our improvement is a finer estimation of the failing probability of the reduction in Waters' original proof relying on artificial abort. We use Bonferroni's inequality, a tunable inequality obtained by cutting off higher order terms from the equality derived by the inclusion-exclusion principle.

Our analysis not only improves the reduction of known constructions but also opens the door for new constructions. While a similar improvement to Waters IBE is possible for the lattice-based IBE by Agrawal, Boneh, and Boyen (Eurocrypt 2010), we can slightly tweak the so-called partitioning function in their construction, achieving $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon/Q), \mathsf{T} + O(Q))$. This is a much better reduction than the previously known $(O(\epsilon^3/Q^2), \mathsf{T} + O(Q))$. We also propose the first VRF with proof and verification key sizes sublinear in the security parameter under the standard d-LIN assumption, while simultaneously improving the reduction cost compared to all prior constructions.

Contents

1 Introduction

1.1 Background

In security proofs for cryptographic primitives, we often face conflicting requirements. For instance, when proving security of a signature scheme, the reduction needs to simulate signatures upon adversary's signing queries and to extract a solution to a computationally hard problem from the forgery. On first glance, such a proof seems to indicate that a reduction can simply simulate an adversary internally: It simulates a forgery instead of running the adversary and extracts the solution from it, contradicting the hardness of the problem. The *partitioning technique* resolves this apparent paradox. The message space is divided into controlled and uncontrolled sets. The reduction can only simulate signatures for controlled messages, while a forgery is only useful if it's for an uncontrolled message. Since a message can only be either controlled or uncontrolled, the paradox is resolved. This technique has been useful outside the simple application of signatures, and in particular, has been central to show *adaptive* security of more advanced primitives such as identity-based encryption (IBE) [Bon01, BB04b, Wat05, ABB10a, CHKP10, ABB10a, CHKP10, ACF14, Yam16, ZCZ16, KY16, Yam17, Kat17, AFL17, ALWW21] and verifiable random function (VRF) with large input spaces [HW09, Jag15, HJ16, Yam17, Kat17, JN19, Koh19, Nie21, JKN21].[1]

A proof relying on the partitioning technique comes in two steps. The *first step* consists of constructing a scheme that secretly partitions the challenge space in controlled and uncontrolled sets during the security proof. This is typically done by implicitly computing a bespoke keyed function F inside the scheme. In the context of signatures, this *partitioning function* $\mathsf{F}(\mathsf{M})$ is secretly computed during the signing algorithm, where $\mathsf{F}(\mathsf{M}) = 1$ (resp. 0) indicates that M is included in the controlled (resp. uncontrolled) set. For the reduction, the probability that the adversarial queries are consistent with the partition made by F needs to be high enough. Specifically, the probability that (i) $\mathsf{F}(\mathsf{M}^{(i)}) = 1$ for all messages $(\mathsf{M}^{(i)})_{i\in[Q]}$ queried to the signing oracle and (ii) $\mathsf{F}(\mathsf{M}^*) = 0$ for the forgery message M^* must be noticeable. Below, we denote this probability as $\gamma(\vec{\mathsf{M}})$, where $\vec{\mathsf{M}} := (\mathsf{M}^{(1)}, \cdots, \mathsf{M}^{(Q)}, \mathsf{M}^*)$. The *second step* is to lower bound the advantage $\epsilon_{\mathsf{proof}}$ of the reduction using the advantage of the adversary ϵ. This step is trivial when reducing a hard search problem to a search type security game (e.g., unforgeability of a signature scheme) as we have a simple lower bound $\epsilon_{\mathsf{proof}} \geq \gamma_{\mathsf{min}}\epsilon$, where $\gamma_{\mathsf{min}} = \min_{\vec{\mathsf{M}}} \gamma(\vec{\mathsf{M}})$. However, such a simple bound no longer holds when reducing a hard decisional problem to a decisional security game, those considered by IBEs and VRFs. Studying the partitioning technique in this non-trivial setting is the main focus of our work.[2]

To the best of our knowledge, there are only two solutions to the second step of the partitioning technique. The first solution originates to Waters [Wat05], who identified this non-triviality when proving security of his IBE. His main observation was that it suffices to *efficiently approximate* $\gamma(\vec{\mathsf{ID}})$ to lower bound $\epsilon_{\mathsf{proof}}$, where we replace $\vec{\mathsf{M}}$ with $\vec{\mathsf{ID}}$ to be consistent with our IBE explanation. Namely, he used the Monte Carlo method to approximate $\gamma(\vec{\mathsf{ID}})$ and completed the reduction using the notorious *artificial abort* step; a counterintuitive step where the reduction sometimes aborts the simulation and outputs a random guess, even if the simulation is successful (i.e., the adversarial queries lie in the correct constrained and unconstrained sets). While this solved the elusive problem of using the partitioning technique for decisional security games, the main caveat was that performing artificial abort incurred a huge runtime loss due to the Monte Carlo method.

[1] Other techniques to achieve adaptive security relying on specific algebraic structures (e.g., dual system encryption) exists. See Sec. 1.3 for more details.

[2] Looking ahead, the difficulty stems from the fact that in a decisionoal security game, the adversary may have a *negative* advantage conditioned on $\vec{\mathsf{M}}$. We refer to Sec. 2.1 for the details.

Denoting the runtime of the reduction and adversary by $\mathsf{T}_{\mathsf{proof}}$ and T, respectively, we have $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon/Q), \mathsf{T} + O(Q^2/\epsilon^2))$, where Q is the number of key queries made by the adversary.[3] The second solution is due to Bellare and Ristenpart [BR09]. They showed that if the value of $\gamma(\vec{\mathsf{ID}})$ for all $\vec{\mathsf{ID}}$ lie in a narrow enough interval, the artificial abort step by Waters can be removed from the reduction. Specifically, the reduction no longer needs to run the costly Monte Carlo method. To satisfy this condition on $\gamma(\vec{\mathsf{ID}})$, they further proposed a new partitioning function F. Altogether, they achieve a better reduction with $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon^2/Q), \mathsf{T}+O(Q))$, shaving off a factor Q in total. However, notice the advantage $\epsilon_{\mathsf{proof}}$ becomes worser than Waters due to the modification they made to the partitioning function F. Importantly, both solutions still have a reduction loss quadratic in ϵ.

Surprisingly, this analysis of $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}})$, i.e., the second step of the partitioning technique, has not seen any improvement for over 15 years. Indeed, all previously cited IBEs [Bon01, BB04b, Wat05, ABB10a, CHKP10, ABB10a, CHKP10, ACF14, Yam16, ZCZ16, KY16, Yam17, Kat17, AFL17, ALWW21] and VRFs [HW09, Jag15, HJ16, Yam17, Kat17, JN19, Koh19, Nie21, JKN21] have proofs based on the partitioning technique that rely either on a Waters-style analysis or a Bellare-Ristenpart-style analysis — most of the improvements come from designing a better partitioning function F with a compatible scheme, i.e., improving the first step of the partitioning technique. This motivates us with the following question:

> *Can we achieve a better reduction cost for proofs based on the partitioning technique? That is, is there a better analysis than those by Waters [Wat05] and Bellare and Ristenpart [BR09]?*

1.2 Our Contributions

In this paper, we answer the above question affirmatively by proposing a new analysis for proofs based on the partitioning technique. Using our analysis, we improve the reduction cost of many of the aforementioned IBEs and VRFs *without* any modification to the construction. For example, Waters IBE can now be proven secure with $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon^{3/2}/Q), \mathsf{T} + O(Q))$, breaking the quadratic dependence on ϵ. We further obtain the same reduction cost for the lattice-based Agrawal-Boneh-Boyen (ABB) IBE [ABB10a], where the known reduction was quite loose, only achieving $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon^3/Q^2), \mathsf{T} + O(Q))$.[4]

Our analysis not only improves the reduction of known constructions but also opens the door for new constructions. Concretely, we construct an IBE and VRF with novel properties.

- By slightly tweaking the ABB IBE construction, we obtain an IBE with a reduction $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = \left(O(\epsilon^{1+\frac{1}{d-1}}/Q), \mathsf{T} + O(Q)\right)$, where $d \geq 3$ is a tunable positive integer that roughly dictates the length of the public parameter. When $d = 3$, we recover the ABB IBE, modulo the small difference in how an identity ID is hashed to matrices. By setting $d = \omega(1)$, we achieve $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon/Q), \mathsf{T} + O(Q))$, which can be thought of as an *ideal* reduction for a partitioning based proof, matching the lower bound for the (black-box) reduction for Waters IBE [HJK12].

- We propose the first VRF achieving sublinear verification key and proof sizes (in the security parameter) under the standard d-LIN assumption. Previous VRFs only achieved this under

[3] Throughout the introduction, we ignore factors only depending on the security parameter λ and focus on the adversarially dependent Q and ϵ.

[4] To be precise, we modify the partitioning function used in ABB-IBE in a superficial manner, so technically speaking, it is no longer an identical scheme (see Sec. 2.7).

non-static q-type assumptions. In fact, we propose two VRFs, where one achieves an $\omega(1)$ proof size at the cost of increasing the verification key size slightly compared to the other. Moreover, the two VRFs enjoy a reduction of $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = \big(O(\epsilon^{1.5}/Q), \mathsf{T} + O(Q)\big)$ and $\big(O(\epsilon^{1+\frac{\mu}{2}}/Q^{\mu}), \mathsf{T} + O(Q)\big)$ for an arbitrary constant $\mu > 1$, respectively. All prior reductions of VRFs with either sublinear verification key or proof sizes only achieve $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = \big(O(\epsilon^{1+\mu}/Q^{\mu}), \mathsf{T} + O(Q)\big)$, or worse. We refer to Table 2 in Sec. 7.5 for the detailed comparison.

At the core of our technical contribution is a new framework for partitioning that interpolates the analysis of Waters and Bellare-Ristenpart in a way that we achieve the best of both worlds. Recall Waters [Wat05] used the naive Monte Carlo method to approximate $\gamma(\vec{\mathsf{ID}})$. While this leads to a good approximation, it suffers from longer runtime of $O(Q^2/\epsilon^2)$. In contrast, Bellare and Ristenpart [BR09] show that if $\gamma(\vec{\mathsf{ID}})$ for all $\vec{\mathsf{ID}}$ lie within a narrow enough interval, the expensive approximation step can be removed. This intuitively requires that a fixed value $\tilde{\gamma}$ can be used as a good enough approximation for $\gamma(\vec{\mathsf{ID}})$ for *all* $\vec{\mathsf{ID}}$. To realize this restrictive condition, they have to change the partitioning function F, leading to a worser advantage $\epsilon_{\mathsf{proof}} = O(\epsilon^2/Q)$.

In our work, we resurrect Waters' artificial abort step, where we approximate $\gamma(\vec{\mathsf{ID}})$ for *each* $\vec{\mathsf{ID}}$, rather than requiring a single approximation $\widetilde{\gamma}$ that works for $\gamma(\vec{\mathsf{ID}})$ for *all* $\vec{\mathsf{ID}}$ as Bellare-Ristenpart. This provides us with greater flexibility in selecting the partitioning function F compared to Bellare-Ristenpart and opens up the potential for achieving a higher advantage $\epsilon_{\mathsf{proof}}$. To this end, we require an improved approximation for $\gamma(\vec{\mathsf{ID}})$ in comparison to Bellare and Ristenpart, as well as an efficient algorithm for computing this approximation in comparison to the Monte-Carlo method by Waters. For a better approximation of $\gamma(\vec{\mathsf{ID}})$, we use Bonferroni's inequality [Bon36], a tunable inequality obtained by cutting of higher order terms from the equality derived by the inclusion-exclusion principle. The evaluation of $\gamma(\mathsf{ID})$ by Bellare and Ristenpart, which uses union bound, can be seen as an application of the special case of Bonferroni's inequality. Now, computing an approximation of $\gamma(\vec{\mathsf{ID}})$ depends on the concrete choice of the partitioning function F. In one case, used by Waters IBE, we need to solve certain counting problem efficiently. For this purpose, we use *generating functions* — a standard tool in enumerative combinatorics but seldom used in cryptography. This part may be of independent interest.

Given that the second step of the partitioning technique remains independent of the underlying primitives (e.g., IBE or VRF) and algebraic structures (e.g., pairings or lattices), we abstract it as a partitioning function *with approximation*, an extension of the partitioning function due to Yamada [Yam17]. We extend the prior definition by augmenting it with an efficient algorithm that estimates $\gamma(\vec{\mathsf{ID}})$. We revisit partitioning functions implicitly used in previous works [Wat05, ABB10a, Lys02], observing that they fit within our abstraction.

Lastly, our new analysis indicates that it is beneficial to choose a partitioning function F that allows to nicely and efficiently approximate $\gamma(\vec{\mathsf{ID}})$. This leads to new ideas to improve the first step of the partitioning technique. For example, we show that by slightly tweaking the partitioning function F used in ABB IBE, we can efficiently compute the Bonferroni's inequality at a much higher order, allowing for a better approximation of $\gamma(\vec{\mathsf{ID}})$. Further details are given in Sec. 2.

1.3 Related Works

Related Works on IBEs. The notion of IBE [Sha84] is introduced as a tool for simplifying the key management in e-mail systmes. The first constructions of IBE are given by [BF01, SOK00] on groups with bilinear maps in the random oracle model. Since then, a large number of IBE schemes have been proposed. We know constructions from quadratic residue [Coc01, BGH07], bilinear

maps [BB04a, BB04b, Wat05, Gen06, Wat09], groups without bilinear map [DG17, BLSV18], from factoring [DG17], lattices [GPV08, CHKP10, ABB10a], and (a strong variant of) learning with parity [BLSV18]. In subsequent works, IBEs with various trade-offs between efficiency, underlying assumptions, and tightness of the reduction have been proposed from the pairings [Lew12, CLL+13, RCS12, JR13, CW13, BKP14, AHY15, GDCC16, CGW17] and from lattices [Yam16, BL16, ZCZ16, KY16, Yam17, Kat17, AFL17, ALWW21, KTY23]. Notably, [BL16] proposes tightly secure IBE from lattices achieving $(\epsilon_{\mathsf{proof}}, \mathsf{T}_{\mathsf{proof}}) = (O(\epsilon), \mathsf{T} + O(Q))$. However, their scheme requires the fully homomorphic evaluation of PRF, which is highly inefficient and requires LWE with super-polynomial modulus.

In the context of pairing-based IBE, the powerful machinery of the dual system encryption methodology has been devised [Wat09, LW10]. To name a few, the technique enables compact public parameters for IBEs [Wat09], (almost) tightly secure IBEs [CW13, BKP14], and adaptive security for primitives beyond IBE [Wat09, LOS+10]. However, there are still many important classes of schemes for which the dual system encryption methodology can not be applied and the partitioning technique is essentially the only option. This includes lattice-based IBEs [ABB10a, CHKP10, Yam16, ZCZ16, KY16, Yam17, Kat17, AFL17, ALWW21], pairing-based IBEs from from computational/decisional bilinear Diffie-Hellman (CBDH/DBDH) problem [Wat05, KY16], pairing IBE with short ciphertext overhead consisting of two group elements [Wat05].

Related Works on VRFs. The notion of VRF is introduced by Micali, Rabin, and Vadhan [MRV99]. Since then, several constructions has been proposed [MRV99, Lys02, Dod03, DY05]. These constructions only allow a polynomial bounded input space, or do not achieve adaptive security without complexity leveraging. The first construction with "all the desired properties" [HJ16], namely, exponentially large input space and a proof of adaptive security under a non-interactive complexity assumption was proposed by Hohenberger and Waters [HW09]. Subsequently, a large number of constructions have been proposed based on pairings [ACF09, HW09, BMR10, ACF14, Jag15, HJ16, Yam17, Kat17, Ros18, JN19, Koh19, Nie21, JKN21] with trade-offs between efficiency, the underlying assumptions, and tightness of the reductions. Notably, Hofheinz and Jager [HJ16] proposed the first VRF with all the desired properties from the standard d-LIN assumption. We also know constructions from general assumptions [Bit17, GHKW17, BGJS17]. We finally note that the dual system encryption methodology has not been successfully applied to the construction of VRF, even in the pairing settings.

Related Works on Partitioning Techniques. Many prior works focus on the first step of the partitioning technique, namely, designing of the partitioning function F and compatible algebraic structures. Concrete examples are for instance the admissible hash function [Lys02, BB04b, CHKP10, FHPS13], Waters hash [Wat05, BR09, HK08, HW09] and its variant [ABB10a, Boy10], and others [Yam16, ZCZ16, Yam17, ALWW21]. These partitioning functions lead to IBEs and VRFs with various tradeoffs between efficiency, underlying assumption, and tightness when combined with suitable algebraic structures. Several works abstract out the algebraic structure that is compatible with the partitioning. In particular, Hofheinz and Kiltz [HK08] introduce the notion of programmable hash functions on pairing groups, which abstracts out the properties of Waters hash [Wat05]. They show new applications along with novel asymptotic analysis. Zhang, Chen, and Zhang [ZCZ16] extend the notion of programmable hash function to the lattice settings. Importantly though, all the above works on IBEs and VRFs rely either on the Waters-style analysis or Bellare-Ristenpart-style analysis to argue the second step of the partitioning technique[5], possibly resulting in a sub-optimal reduction.

[5]The work by Hofheinz and Kiltz [HK08] does not explicitly consider an application to IBEs. However, we can

Road Map. We first provide the overview of our techniques in Sec. 2. Preliminalies are in Sec. 3 due to page limitations. In Sec. 4, we provide a general theorem enabling a finer grained analysis of the artificial abort paradigm decoupled from the underlying primitives and algebraic structures. In Sec. 5, we show our new analysis of the new and existing partitioning functions. In Sec. 6, we apply the tools developed in Sec. 4 and 5 to IBEs, e.g., the Waters IBE, the ABB IBE, and a variant of ABB IBE, and show a tighter security. In Sec. 7, we propose a new VRF scheme achieving the best asymptotic space efficiency and tighter security under a standard assumption.

2 Technical Overview

We provide an overview of our results. In Sec. 2.1, we review why a naive proof using the partitioning technique fails. While this is agnostic to the specific application, we use Waters IBE [Wat05] as a representative example. In Sec. 2.2 and 2.3, we explain how Waters resolved the problem using the artificial abort step and see that it suffices to estimate $\gamma(\vec{\mathsf{ID}})$ with certain accuracy. In Sec. 2.4, we explain the reduction by Bellare and Ristenpart [BR09] from this perspective. In Sec. 2.5, we explain the main idea behind our improved reduction for Waters IBE. In Sec. 2.6, we explain our new estimation algorithm for $\gamma(\vec{\mathsf{ID}})$. In Sec. 2.7, we shift our focus and consider partitioning technique in the lattice setting. We then explain that we can improve the reduction cost of ABB IBE and propose a variant of it with an even better reduction cost. Finally, in Sec. 2.8, we shift our focus again and consider a new partitioning technique based on substring matching.

2.1 The Difficulty

Let us review the proof of Waters IBE [Wat05] based on the partitioning technique and observe the non-triviality of it. In his proof, the reduction algorithm for DBDH perfectly simulates the security game for an adversary A against the IBE scheme until it reaches the point where it cannot continue the simulation anymore and has to abort the reduction. The probability that the simulation is not successful only depends on the sequence of identities $\vec{\mathsf{ID}} = (\mathsf{ID}^*, \mathsf{ID}^{(1)}, \ldots, \mathsf{ID}^{(Q)})$, where $\mathsf{ID}^* \in \{0,1\}^\ell$ is the challenge identity for which the challenge ciphertext is generated and $\mathsf{ID}^{(1)}, \ldots, \mathsf{ID}^{(Q)} \in \{0,1\}^\ell$ are identities for which key queries were made. Let us analyze a naive reduction that outputs the same bit as A when the simulation is successful and outputs a random bit when the simulation fails.

We denote the advantage of the adversary A by ϵ, the probability that A makes the sequence of queries $\vec{\mathsf{ID}}$ by $p(\vec{\mathsf{ID}})$, and its advantage conditioned on the sequence of queries $\vec{\mathsf{ID}}$ by $\epsilon(\vec{\mathsf{ID}})$. We have

$$\frac{1}{2} + \epsilon = \sum_{\vec{\mathsf{ID}}} p(\vec{\mathsf{ID}}) \left(\frac{1}{2} + \epsilon(\vec{\mathsf{ID}}) \right) = \frac{1}{2} + \sum_{\vec{\mathsf{ID}}} p(\vec{\mathsf{ID}}) \epsilon(\vec{\mathsf{ID}}),$$

where the sum is taken over all possible $\vec{\mathsf{ID}}$. Denoting the probability of the simulation being successful by $\gamma(\vec{\mathsf{ID}})$,[6] the advantage of the reduction algorithm against DBDH can be evaluated

use their framework in the context of IBE and this requires the heavy artificial abort step in the reduction.

[6] We differentiate "not aborting" and "simulation being successful", since we will later introduce artificial abort, where the simulator aborts even if the simulation is successful. We note that in the naive reduction described here, this distinction is irrelevant.

as

$$\sum_{\vec{\mathsf{ID}}} p(\vec{\mathsf{ID}}) \left(\gamma(\vec{\mathsf{ID}}) \left(\frac{1}{2} + \epsilon(\vec{\mathsf{ID}}) \right) + \frac{1 - \gamma(\vec{\mathsf{ID}})}{2} \right) - \frac{1}{2} = \sum_{\vec{\mathsf{ID}}} \gamma(\vec{\mathsf{ID}}) p(\vec{\mathsf{ID}}) \epsilon(\vec{\mathsf{ID}}). \tag{1}$$

In Waters' proof, it is shown that $\gamma(\vec{\mathsf{ID}}) \geq 1/\mathsf{poly}$ for all possible $\vec{\mathsf{ID}}$. It is tempting to conclude the proof by claiming the above advantage is non-negligible conditioned on ϵ being non-negligible. However, this intuition turns out to be false and this is precisely the reason why artificial abort was introduced in [Wat05]. As an illustrating example, consider an adversary who yields only two types of query sequences $\vec{\mathsf{ID}}_A$ and $\vec{\mathsf{ID}}_B$. We further assume $p(\vec{\mathsf{ID}}_A) = p(\vec{\mathsf{ID}}_B) = 1/2$, $\gamma(\vec{\mathsf{ID}}_A) = 1/3$, $\gamma(\vec{\mathsf{ID}}_B) = 2/3$, $\epsilon(\vec{\mathsf{ID}}_A) = 2/5$, and $\epsilon(\vec{\mathsf{ID}}_B) = -1/5$. Even though the adversary A has advantage $1/10$, the advantage of the reduction algorithm is 0, meaning that it guesses the challenge bit no better than randomly guessing.

The reason why the above problem occurs is that $\epsilon(\vec{\mathsf{ID}})$ can be negative for some $\vec{\mathsf{ID}}$. When the "weight" on $\epsilon(\vec{\mathsf{ID}})$ changes from $p(\vec{\mathsf{ID}})$ to $p(\vec{\mathsf{ID}})\gamma(\vec{\mathsf{ID}})$ due to the failure of the simulation, the negative $\epsilon(\vec{\mathsf{ID}})$ may be amplified to cancel out the positive $\epsilon(\vec{\mathsf{ID}}')$, rendering the total sum being negligible. It is worth highlighting that this is exactly why partitioning based proofs are easier for search type games since $\epsilon(\vec{\mathsf{ID}}) \geq 0$ is guaranteed by definition (see Footnote 2).

2.2 Artificial Abort

We then move to explain in several steps how Waters [Wat05] resolved the above problem by introducing the artificial abort step. First, observe that if $\gamma(\vec{\mathsf{ID}}) = \gamma$ holds for some fixed $\gamma \geq 1/\mathsf{poly}$ for all $\vec{\mathsf{ID}}$, the above naive reduction works. This is because we have the following which is non-negligible:

$$\sum_{\vec{\mathsf{ID}}} \gamma(\vec{\mathsf{ID}}) p(\vec{\mathsf{ID}}) \epsilon(\vec{\mathsf{ID}}) = \gamma \sum_{\vec{\mathsf{ID}}} p(\vec{\mathsf{ID}}) \epsilon(\vec{\mathsf{ID}}) = \gamma\epsilon.$$

We then move to the more realistic setting where $\gamma(\vec{\mathsf{ID}})$ varies with $\vec{\mathsf{ID}}$. Here, we still assume that $\gamma(\vec{\mathsf{ID}}) \geq \gamma_{\min}$ holds for all $\vec{\mathsf{ID}}$ and for some fixed $\gamma_{\min} \geq 1/\mathsf{poly}$. For the sake of explanation, we also introduce a simplifying assumption that $\gamma(\vec{\mathsf{ID}})$ can be computed efficiently given $\vec{\mathsf{ID}}$. In this setting, we can make the reduction work by introducing an additional abort step (i.e., artificial abort). Namely, after having successfully completed the simulation against the adversary, the simulator evaluates $\gamma(\vec{\mathsf{ID}})$ based on the sequence of queries $\vec{\mathsf{ID}}$. It then aborts with probability $1 - \gamma_{\min}/\gamma(\vec{\mathsf{ID}})$ and outputs a random bit. Then, the probability of the simulation not aborting is the same for all $\vec{\mathsf{ID}}$, namely, $\gamma_{\min}$. We therefore can use the above analysis to conclude that the advantage of the final adversary is $\gamma_{\min}\epsilon$, which is non-negligible.

However, in reality, we do not know how to compute $\gamma(\vec{\mathsf{ID}})$ efficiently. What Waters [Wat05] did instead is to approximate the value of $\gamma(\vec{\mathsf{ID}})$ by the Monte Carlo method. The simulator repeatedly chooses simulation randomness, sees if each randomness leads to a successful simulation, and uses the fraction of randomness that leads to a successful simulation as an approximation for $\gamma(\vec{\mathsf{ID}})$. We do not give details of the analysis by [Wat05] further, since it is irrelevant to the overview. We just note that the Monte Carlo method is expensive and the approximation needs time proportional to $O(Q^2/\epsilon^2)$ to compute.

2.3 Accuracy of Approximation

Let us discuss how the accuracy of the approximation $\gamma(\vec{\mathsf{ID}})$ affects the reduction. We note that our explanation here is different from the analysis by [Wat05] and is an extension of the analysis

by Bellare and Ristenpart [BR09]. For the sake of easier exposition, we first show our general analysis and then explain the analysis by [BR09] as a special case. Let us assume that $\gamma(\vec{\mathsf{ID}})$ can be approximated efficiently and deterministically. We denote the approximation for $\gamma(\vec{\mathsf{ID}})$ by $\widetilde{\gamma}(\vec{\mathsf{ID}})$. At the end of the simulation, the reduction algorithm aborts and outputs a random bit with probability $1-\gamma_{\min}/\widetilde{\gamma}(\vec{\mathsf{ID}})$, with the intention of making the abort probability as independent of $\vec{\mathsf{ID}}$ as possible. We then discuss the advantage of the adversary. Since we have just changed the abort probability, we can see that the advantage of the reduction algorithm is obtained by replacing $\gamma(\vec{\mathsf{ID}})$ in Eq. (1) with $\gamma_{\min}\cdot\gamma(\vec{\mathsf{ID}})/\widetilde{\gamma}(\vec{\mathsf{ID}})$, which is the probability that the reduction algorithm does not abort conditioned on the sequence of the identities in the simulation is $\vec{\mathsf{ID}}$. Namely, the advantage is

$$\sum_{\vec{\mathsf{ID}}}\gamma_{\min}\cdot\left(\frac{\gamma(\vec{\mathsf{ID}})}{\widetilde{\gamma}(\vec{\mathsf{ID}})}\right)\cdot p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})=\gamma_{\min}\cdot\left(\sum_{\vec{\mathsf{ID}}}(1+\Delta(\vec{\mathsf{ID}}))\cdot p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})\right),$$

where we define $\Delta(\vec{\mathsf{ID}}) := \gamma(\vec{\mathsf{ID}})/\widetilde{\gamma}(\vec{\mathsf{ID}})-1$. In the following, we will argue that if $\Delta(\vec{\mathsf{ID}})$ is sufficiently small, we can give a useful lower bound for the above quantity. Toward this goal, we assume $-\overline{\Delta}\le\Delta(\vec{\mathsf{ID}})\le\overline{\Delta}$ and continue the analysis. We have

$$\begin{aligned}\sum_{\vec{\mathsf{ID}}}(1+\Delta(\vec{\mathsf{ID}}))\cdot p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}}) &= \epsilon+\sum_{\vec{\mathsf{ID}}}\Delta(\vec{\mathsf{ID}})p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})\\ &\ge \epsilon+\sum_{\vec{\mathsf{ID}}\text{ s.t. }\epsilon(\vec{\mathsf{ID}})\ge 0}(-\overline{\Delta})\cdot p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})+\sum_{\vec{\mathsf{ID}}\text{ s.t. }\epsilon(\vec{\mathsf{ID}})<0}\overline{\Delta}\cdot p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})\\ &\ge \epsilon-2\overline{\Delta},\end{aligned}$$

where the first line uses $\sum_{\vec{\mathsf{ID}}}p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})=\epsilon$ and the third line uses $\sum_{\vec{\mathsf{ID}}\text{ s.t. }\epsilon(\vec{\mathsf{ID}})\ge 0}p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})\le 1$ and $\sum_{\vec{\mathsf{ID}}\text{ s.t. }\epsilon(\vec{\mathsf{ID}})<0}p(\vec{\mathsf{ID}})\epsilon(\vec{\mathsf{ID}})\ge -1$. This analysis shows that if $\overline{\Delta}<\epsilon/3$, we have that the overall advantage of the reduction algorithm is at least $\gamma_{\min}\epsilon/3$, which is non-negligible. Recalling the definition of $\overline{\Delta}$, this means that for the reduction to work, it suffices to approximate $\gamma(\vec{\mathsf{ID}})$ within an additive error no greater than $\gamma_{\min}\epsilon/3$.

We then move to explain the idea of Bellare and Ristenpart [BR09] as a special case of the above reduction strategy. We can regard their reduction algorithm as a special case of the above reduction, where the approximation $\widetilde{\gamma}(\vec{\mathsf{ID}})$ for $\gamma(\vec{\mathsf{ID}})$ is always set to be $\gamma_{\min}$, regardless of $\vec{\mathsf{ID}}$. This means that the reduction algorithm never artificially aborts, since $1-\gamma_{\min}/\widetilde{\gamma}(\vec{\mathsf{ID}})=0$. As we have discussed above, we need to have $\overline{\Delta}<\epsilon/3$, which implies that $(1-\epsilon/3)\gamma_{\min}\le\gamma(\vec{\mathsf{ID}})\le(1+\epsilon/3)\gamma_{\min}$ for all $\vec{\mathsf{ID}}$. They achieve this condition by finding a clever choice of parameters. We defer the detail to the next subsection.

2.4 Simulation Method of Bellare and Ristenpart [BR09]

To explain their idea, we have to dive into details of how $\gamma(\vec{\mathsf{ID}})$ is defined for a sequence of queries $\vec{\mathsf{ID}}$. Recall that in the security proof using the partitioning technique, we divide the identity space into controlled and uncontrolled sets based on a secret randomness K. In the security proof by [Wat05, BR09], they divide the identity space by the following (partitioning) function

$$\mathsf{F}_{\mathsf{Wat}}(K,\mathsf{ID})=\begin{cases}0 & \text{(Meaning ``uncontrolled'')} \quad \text{if} \quad K_0+\sum_{i:\mathsf{ID}_i=1}K_i=0\\ 1 & \text{(Meaning ``controlled'')} \quad \text{if} \quad K_0+\sum_{i:\mathsf{ID}_i=1}K_i\neq 0.\end{cases}, \tag{2}$$

where $K = (K_0, K_1 \ldots, K_\ell)$ is the secret randomness chosen as $K_0 \xleftarrow{\$} [-\ell N, 0]$ and $K_i \xleftarrow{\$} [0, N]$ for $i \in [\ell]$, ℓ denotes the binary length of identities, and ID_i denotes the i-th bit of an identity $\mathsf{ID} \in \{0,1\}^\ell$. We will explain how they determine the parameter N later. Recall that $\gamma(\vec{\mathsf{ID}})$ is the probability that the simulation is successful. Namely, this is the probability that ID^* falls into the uncontrolled set and all of $\mathsf{ID}^{(1)}, \ldots, \mathsf{ID}^{(Q)}$ fall into the controlled set. Denoting the event that $\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{ID}^*) = 0$ holds by E^* and the event that $\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{ID}^{(j)}) = 0$ holds for $j \in [Q]$ by $\mathsf{E}^{(j)}$, we have

$$\begin{aligned}\gamma(\vec{\mathsf{ID}}) &= \Pr[\mathsf{E}^* \wedge \neg\mathsf{E}^{(1)} \cdots \wedge \neg\mathsf{E}^{(Q)}] \\ &= \Pr[\mathsf{E}^*] - \Pr[\mathsf{E}^* \wedge (\mathsf{E}^{(1)} \vee \cdots \vee \mathsf{E}^{(Q)})] \\ &= \Pr[\mathsf{E}^*] - \Pr[(\mathsf{E}^* \wedge \mathsf{E}^{(1)}) \vee \cdots \vee (\mathsf{E}^* \wedge \mathsf{E}^{(Q)})], \end{aligned} \tag{3}$$

where the probability is taken over the choice of K.

Since it is straightforward to see $\Pr[\mathsf{E}^*] = 1/(\ell N + 1)$, getting approximation for $\gamma(\vec{\mathsf{ID}})$ boils down to getting approximation for $\Pr[(\mathsf{E}^* \wedge \mathsf{E}^{(1)}) \vee \cdots \vee (\mathsf{E}^* \wedge \mathsf{E}^{(Q)})]$. They use the union bound to upper bound the term and give a trivial lower bound 0, which results in the following inequality:

$$\Pr[\mathsf{E}^*] - \underbrace{\sum_{j \in [Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]}_{= \text{Approximation error}} \leq \gamma(\vec{\mathsf{ID}}) \leq \Pr[\mathsf{E}^*]. \tag{4}$$

Recall that they use fixed $\gamma_{\min}$ as an approximation for $\gamma(\vec{\mathsf{ID}})$ for all $\vec{\mathsf{ID}}$ and for the reduction to work, the approximation should be within additive error of $\gamma_{\min}\epsilon/3$. To achieve this guarantee, they adjust the parameter so that the approximation error term $\sum_{j \in [Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$ is as small as possible. Let us introduce the parameter δ, which is defined as $\delta := \Pr[\mathsf{E}^*]$. We can easily see that δ can be controlled by adjusting the parameter N and $\Pr[\mathsf{E}^{(j)}] = \delta$ for $j \in [Q]$ holds. For the sake of simplicity of the explanation, we introduce an oversimplifying assumption that these events are pair-wise independent, meaning that $\Pr[\mathsf{E}^{(j)} \wedge \mathsf{E}^*] = \delta^2$ and $\Pr[\mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}] = \delta^2$. We then upper bound the error term as

$$\sum_{j \in [Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] \leq Q\delta^2.$$

What remains is to choose $\gamma_{\min}$ and δ so that $\gamma_{\min} \leq \delta - Q\delta^2$ and $Q\delta^2 \leq \gamma_{\min}\epsilon/3$ hold. The latter inequality implies $Q\delta^2 \ll \gamma_{\min}$ and the former then implies that we can take $\gamma_{\min} = \delta/2$ for example. Then, the latter implies $Q\delta^2 \leq \delta\epsilon/6$, which in turn implies $\delta \leq \epsilon/6Q$. Therefore, we set $\delta = \Theta(\epsilon/Q)$ and then the advantage of the reduction algorithm is $\gamma_{\min}\epsilon/3 = \Theta(\delta\epsilon) = \Theta(\epsilon^2/Q)$.[7]

2.5 More Sophisticated Approximation

Our idea to improve the reduction algorithms of previous works [Wat05, BR09] is to approximate $\gamma(\vec{\mathsf{ID}})$ by a more sophisticated analysis. In particular, we approximate the term $\Pr[(\mathsf{E}^* \wedge \mathsf{E}^{(1)}) \vee \cdots \vee (\mathsf{E}^* \wedge \mathsf{E}^{(Q)})]$ in Eq. (3) by Bonferroni's inequalities[8], rather than the union bound. Namely,

[7] Due to the simplifying assumption, the bound here does not exactly correspond to that given in [BR09]. More formally, we have an extra cost of $O(1/\ell)$ in the final advantage. Similar remark applies to other analyses that appear in the overview.

[8] Bonferroni's inequalities are the inequalities obtained by cutting off higher order terms from the equality derived from inclusion-exclusion principle. See Lemma 1 for the formal statement.

we have

$$\sum_{j\in[Q]} \Pr[\mathsf{E}_2^{(j)}] - \sum_{1\le j<k\le Q} \Pr[\mathsf{E}_2^{(j)} \wedge \mathsf{E}_2^{(k)}] \le \Pr[\mathsf{E}_2^{(1)} \vee \cdots \vee \mathsf{E}_2^{(Q)}] \le \sum_{j\in[Q]} \Pr[\mathsf{E}_2^{(j)}],$$

where we denote $\mathsf{E}_2^{(j)} := \mathsf{E}^* \wedge \mathsf{E}^{(j)}$ for notational convenience in the above. Plugging the above equation into Eq. (3), we obtain

$$\Pr[\mathsf{E}^*] - \sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] \le \gamma(\vec{\mathsf{ID}}) \le \Pr[\mathsf{E}^*] - \sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] + \underbrace{\sum_{1\le j<k\le Q} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}]}_{=\text{Approximation error}}, \quad (5)$$

where we use $\Pr[\mathsf{E}_2^{(j)} \wedge \mathsf{E}_2^{(k)}] = \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}]$. We then use $\Pr[\mathsf{E}^*] - \sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$ as the approximation for $\gamma(\vec{\mathsf{ID}})$, i.e., $\widetilde{\gamma}(\vec{\mathsf{ID}}) := \Pr[\mathsf{E}^*] - \sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$. While for this to be useful, we have to show that we can efficiently compute $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$, we simply assume this is possible and defer the detail to Sec. 2.6. Now, observe that the approximation error can be bounded by $\sum_{1\le j<k\le Q} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}]$. We analyze this term by introducing again an oversimplifying assumption that the events $\mathsf{E}^*, \mathsf{E}^{(1)}, \ldots, \mathsf{E}^{(Q)}$ are 3-wise independent, meaning that any conjunction of 3 of them happens with probability δ^3. Using this, we bound the above approximation error term by $Q(Q-1)\delta^3/2 \le Q^2\delta^3$. By our condition on the approximation error, we need to satisfy

$$Q^2\delta^3 \le \gamma_{\min}\epsilon/3.$$

We also have to set $\gamma_{\min}$ so that it is smaller than the leftmost term in Eq. (5). We have $\sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] = Q\delta^2$ by our assumption of pairwise independence and thus the condition is equivalent to

$$\gamma_{\min} \le \delta - Q\delta^2.$$

By a similar analysis explained in the previous subsection, we can take $\gamma_{\min} = \delta/2$. We then have $Q^2\delta^3 \le \delta\epsilon/6$, resulting in $\delta \le \epsilon^{1/2}/6Q$. By setting $\delta = \Theta(\epsilon^{1/2}/Q)$, the advantage of the reduction algorithm becomes $\gamma_{\min}\epsilon/3 = \Theta(\delta\epsilon) = \Theta(\epsilon^{1.5}/Q)$, improving the result of [BR09]. The reason for this improvement is our fine-grained approximation of $\gamma(\vec{\mathsf{ID}})$ compared to [BR09] based on Bonferroni's inequality. By representing the approximation error as a higher order polynomial of δ, we can chose a larger δ (i.e., $\Theta(\epsilon^{0.5}/Q)$ as opposed to $\Theta(\epsilon/Q)$), leading to a better advantage.

2.6 Computing the Probability Efficiently

Two things are missing from the above explanation. First, in the above analysis, we assumed that the events E^*, $\mathsf{E}^{(1)}, \ldots, \mathsf{E}^{(Q)}$ are 3-wise independent. Unfortunately, this assumption is not true. However, we can show that $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}] = \Theta(\ell^2\delta^3)$ for $j \ne k$, which is still useful for the analysis. We defer the details on how to prove this to the main body of the paper. The other more important detail missing from the above explanation is how to compute $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$ efficiently for $j \in [Q]$. The rest of this subsection will be devoted on explaining how to do it. Let us define $S := \{i \in [\ell] : \mathsf{ID}_i^* = 1\}$ and $T := \{i \in [\ell] : \mathsf{ID}_i^{(j)} = 1\}$. Then, by the definition of E^* and $\mathsf{E}^{(j)}$, we have

$$\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] = \Pr\left[K_0 + \sum_{i\in S} K_i = 0 \ \wedge \ K_0 + \sum_{i\in T} K_i = 0\right] = \frac{1}{\ell N + 1} \cdot \Pr\left[\sum_{i\in S} K_i = \sum_{i\in T} K_i\right], \quad (6)$$

where the probability is taken over the randomness of $K_0 \xleftarrow{\$} [-\ell N, 0]$ and $K_i \xleftarrow{\$} [0, N]$ for $i \in [\ell]$. Without loss of generality, we can assume that $S \cap T = \emptyset$. Furthermore, we can assume that $S = [n_S]$ and $T = [n_S + 1, n_S + n_T]$, where $n_S = \#S$ and $n_T = \#T$ with $n_S \leq n_T$. Toward computing the above probability, we introduce a function R, defined as

$$R_n(\alpha) := \# \left\{ 0 \leq K_i \leq N \ : \ \sum_{i \in [n]} K_i = \alpha \right\}.$$

Using the notation, we continue the analysis from Eq. (6). We have

$$\begin{aligned}
& \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] \\
= \ & \frac{1}{(\ell N + 1)(N+1)^{n_S + n_T}} \cdot \# \left\{ K_i \in [0, N] \text{ for } i \in [n_S + n_T] \ : \ \sum_{i \in [n_S]} K_i = \sum_{i \in [n_T]} K_i \right\} \\
= \ & \frac{1}{(\ell N + 1)(N+1)^{n_S + n_T}} \cdot \sum_{\alpha=0}^{n_S N} \# \left\{ K_i \in [0, N] \text{ for } i \in [n_S + n_T] \ : \ \sum_{i \in [n_S]} K_i = \sum_{i \in [n_T]} K_i = \alpha \right\} \\
= \ & \frac{1}{(\ell N + 1)(N+1)^{n_S + n_T}} \cdot \sum_{\alpha=0}^{n_S N} R_{n_S}(\alpha) R_{n_T}(\alpha).
\end{aligned}$$

At this point, the problem of estimating the probability boils down to the problem of computing the summation $\sum_{\alpha=0}^{n_S N} R_{n_S}(\alpha) R_{n_T}(\alpha)$. We emphasize that the algorithm needs to run in at most poly-logarithmic time in N. Otherwise, our final reduction algorithm will add an additive overhead $Q \cdot \mathsf{poly}(N) = Q \cdot \mathsf{poly}(Q, 1/\epsilon)$ to the running time, which ruins the merit of having a larger distinguishing advantage for the reduction algorithm compared to [BR09].

The most natural approach for solving the problem would be to take a dynamic programming approach to compute $R_n(\alpha)$ for $n \in \{n_S, n_T\}$ and $\alpha \in [0, n_T]$ and then compute the summation. This approach is problematic in two-folds: First, computing $R_n(\alpha)$ by dynamic programming requires $\mathsf{poly}(N)$ time, which is too slow. Furthermore, even if $R_\ell(\alpha)$ can be computed efficiently, we have to compute the summation of $n_S N$ terms, which requires $\mathsf{poly}(N)$ time if we follow the straightforward approach. Luckily, there is an elegant solution to the first problem of computing $R_n(\alpha)$ efficiently using the powerful machinery of *generating functions* [Wil05], which is a standard tool in enumerative combinatrics. Furthermore, we can show the equation

$$\sum_{\alpha=0}^{n_S N} R_{n_S}(\alpha) R_{n_T}(\alpha) = R_{n_S + n_T}(n_T N) \tag{7}$$

again using generating functions and therefore the summation can be computed efficiently. We defer the detail of how to compute $R_\ell(\alpha)$ to the main body and explain how to prove the equation here.

Before the proof, let us define a useful notation. For a polynomial $f(\mathsf{Z}) = \sum_i a_i \mathsf{Z}^i$ with Z being indeterminate, we denote $[\mathsf{Z}^j] f(\mathsf{Z})$ as the j-th coefficient of $f(\mathsf{Z})$, namely, a_j. We then observe that $R_n(\alpha)$ equals to $[\mathsf{Z}^\alpha](1 + \mathsf{Z} + \mathsf{Z}^2 + \cdots \mathsf{Z}^N)^n$. This can be seen by expanding the multiplication and observing that to yield the term Z^α, we have to choose Z^{K_i} from the i-th factor so that their sum $K_1 + \cdots K_N$ equals to α. We also observe that $R_n(\alpha) = R_n(nN - \alpha)$, which can be seen by comparing the coefficients of the left and right hands of the equality $(1 + \mathsf{Z} + \mathsf{Z}^2 + \cdots \mathsf{Z}^N)^n =$

$\mathsf{Z}^{nN}(1+\mathsf{Z}^{-1}+\cdots\mathsf{Z}^{-N})^n$. We finally observe that for polynomials $f(\mathsf{Z})$ and $g(\mathsf{Z})$ and an integer n with $n \geq \deg(f)$, we have

$$[\mathsf{Z}^n](f(\mathsf{Z})\cdot g(\mathsf{Z})) = \sum_{i=0}^{\deg(f)} [\mathsf{Z}^i]f(\mathsf{Z})\cdot[\mathsf{Z}^{n-i}]g(\mathsf{Z}).$$

Equipped with the observations, we are now ready to prove Eq. (7). We have:

$$\begin{aligned}
\sum_{\alpha=0}^{n_S N} R_{n_S}(\alpha)R_{n_T}(\alpha) &= \sum_{\alpha=0}^{n_S N} R_{n_S}(\alpha)R_{n_T}(n_T N-\alpha)\\
&= \sum_{\alpha=0}^{n_S N} [\mathsf{Z}^\alpha](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^{n_S}\cdot[\mathsf{Z}^{n_T N-\alpha}](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^{n_T}\\
&= [\mathsf{Z}^{n_T N}](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^{n_S+n_T}\\
&= R_{n_S+n_T}(n_T N)
\end{aligned}$$

as desired.

2.7 Partitioning for Lattices

From here on, we shift our focus and analyze different partitioning strategies. Let us start with a variant of $\mathsf{F}_{\mathsf{Wat}}$ defined in Eq. (2). While the partitioning strategy specified by the function $\mathsf{F}_{\mathsf{Wat}}$ can in principle be used in the lattice setting, it requires a super-polynomial size modulus q for the underlying scheme, since q should be larger than the parameter N, which is polynomially related to Q (and $1/\epsilon$). To refrain from using a superpolynomial modulus q, Boyen [Boy10] proposed a variant of Waters' partitioning function suitable for the lattice setting, later used for proving the security of ABB IBE [ABB10a]. As we explain in Sec. 5.4, our formal analysis reveals that their analysis suffers from a large reduction loss of $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^3/Q^2)$. We show that a more natural adaptation of the Waters' partitioning function to the lattice setting gives us a reduction with $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^2/Q)$, even with the Bellare-Ristenpart-style analysis. This variant is essentially identical to Boyen's partitioning function but fixing some superfluous components (see Footnote 10). Importantly, this is only a superficial difference and keeps the efficiency of the original ABB IBE unchanged. We then show that this can be further improved to $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^{1.5}/Q)$ by our analysis using Bonferroni's inequality. Lastly, with a more noticeable tweak to the partitioning function, we can achieve $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^{1+1/d}/Q)$ for an arbitrary $d > 2$ or even $\gamma_{\mathsf{min}}\epsilon = O(\epsilon/Q)$, where this tweak results in slightly modifying the ABB IBE.

More concretely, we define our partitioning function $\mathsf{F}_{\mathsf{ParWat}}(K,\mathsf{x})$ as follows:

$$\mathsf{F}_{\mathsf{ParWat}}(K,\mathsf{ID}) = \begin{cases} 0 & K_0 + \sum_{i:\mathsf{ID}_i=1} K_i = \mathbf{0}_c \pmod q \\ 1 & \text{otherwise} \end{cases},$$

where $K = (K_0, K_1, \ldots, K_\ell) \in (\mathbb{Z}_q^c)^\ell$ and c is a parameter that will be defined later. $K_0, K_1, \ldots, K_\ell$ are chosen uniformly at random from $\mathbb{Z}_q^c$. We note that here, q is a small polynomially bounded prime.

Bellare-Ristenpart-style analysis. We then analyze $\gamma(\vec{\mathsf{ID}})$. Let us start with a Bellare-Ristenpart-style analysis, where we use a fixed value γ_{min} for the estimation of $\gamma(\mathsf{ID})$. Denoting the event that $\mathsf{F}_{\mathsf{ParWat}}(K,\mathsf{ID}^*) = 0$ holds by E^* and the event that $\mathsf{F}_{\mathsf{ParWat}}(K,\mathsf{ID}^{(j)}) = 0$ holds

for $j \in [Q]$ by $\mathsf{E}^{(j)}$, Eq. (4) can be shown to hold by the same analysis as in Sec. 2.4. We then proceed to bound the error term $\sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$. While this requires a bit of work for the case of $\mathsf{F}_{\mathsf{Wat}}$, it is straightforward here. Concretely, we have

$$\Pr[\mathsf{E}^*] = \frac{1}{q^c}, \qquad \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] = \frac{1}{q^{2c}}$$

for all $j \in [Q]$. Here, the former equation is straightforward to see by the fact that K_0 is distributed uniformly at random over $\mathbb{Z}_q^c$. To see the latter equation, we observe

$$K_0 + \sum_{i:\mathsf{ID}_i=1} K_i = (1, \mathsf{ID}) \cdot (K_0^\top, K_1^\top, \ldots, K_\ell^\top)^\top,$$

where we regard $\mathsf{ID} \in \{0,1\}^\ell$ as a row vector with dimension ℓ and $(K_0^\top, K_1^\top, \ldots, K_\ell^\top)^\top \in \mathbb{Z}_q^{(\ell+1)\times c}$ is a matrix obtained by regarding each K_i as a row vector and concatenating them vertically. When ID^* and $\mathsf{ID}^{(j)}$ are distinct, $(1, \mathsf{ID}^*)$ and $(1, \mathsf{ID}^{(j)})$ are linearly independent and thus the pair $(K_0 + \sum_{i:\mathsf{ID}_i^*=1} K_i, K_0 + \sum_{i:\mathsf{ID}_i^{(j)}=1} K_i)$ are distributed uniformly at random over $\mathbb{Z}_q^{2c}$, implying the above equation.

From the above analysis, we can see that the error term in Eq. (4) can be bounded by $Q \cdot q^{-2c}$. It remains to choose γ_{min} and c so that $\gamma_{\mathsf{min}} \le q^{-c} - Qq^{-2c}$ and $Q \cdot q^{-2c} \le \gamma_{\mathsf{min}}\epsilon/3$ hold. Combining these inequalities, we have $Q \cdot q^{-2c} \le q^{-c}\epsilon/3$. To satisfy this, we choose $c = \log_q(3Q/\epsilon)$, which leads to the reduction cost $\gamma_{\mathsf{min}}\epsilon = \Theta(\epsilon^2/Q)$.

Our improved analysis. We then move to explain our finer-grained analysis using Bonferroni's inequality. Now, by the same analysis as Sec. 2.5 using Bonferroni's inequality, we can derive Eq. (5). We then set $\widetilde{\gamma}(\vec{\mathsf{ID}}) := \Pr[\mathsf{E}^*] - \sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] = q^{-c} - Qq^{-2c}$. Unlike Sec. 2.5, we can directly compute $\widetilde{\gamma}(\vec{\mathsf{ID}})$. We then bound the error term $\sum_{j,k} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}]$ in Eq. (5). We have

$$\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}] = q^{-3c}$$

for each j, k, since we can prove that the vectors $(1, \mathsf{ID}^*)$, $(1, \mathsf{ID}^{(j)})$, and $(1, \mathsf{ID}^{(k)})$ are linearly independent for mutually distinct ID^*, $\mathsf{ID}^{(j)}$, and $\mathsf{ID}^{(k)}$. This allows us to bound the error term by Qq^{-3c}. It remains to choose γ_{min} and c so that $\gamma_{\mathsf{min}} \le q^{-c} - Qq^{-2c}$ and $Q^2 \cdot q^{-3c} \le \gamma_{\mathsf{min}}\epsilon/3$ hold. Combining these inequalities, we have $Q \cdot q^{-3c} \le q^{-c}\epsilon/3$. To satisfy this, we choose $c = \log_q(3Q/\sqrt{\epsilon})$, which leads to the reduction cost $\gamma_{\mathsf{min}}\epsilon = \Theta(\epsilon^{1.5}/Q)$. This improves the bound based on the Bellare-Ristenpart-style analysis by a factor of $\epsilon^{1/2}$.

Going beyond $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^{1.5}/Q)$. A natural question would be whether we can go beyond $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^{1.5}/Q)$ using Bonferroni's inequality with higher order terms. This could be possible if we had $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j_1)} \wedge \cdots \wedge \mathsf{E}^{(j_{d-1})}] = q^{-cd}$ for $d \ge 4$. However, unfortunately, this does not hold already for $d = 4$. We therefore change the function a bit so that

$$\mathsf{F}_{\mathsf{ParWat}}(K, \mathsf{ID}) = \begin{cases} 0 & K_0 + \sum_{i:h_{d\text{-wise}}(\mathsf{ID})_i=1} K_i = \mathbf{0}_c \pmod q \\ 1 & \text{otherwise} \end{cases},$$

where the only change we add is that we hash the identity by a hash function $h_{d\text{-wise}} : \{0,1\}^\ell \to \{0,1\}^{L_d}$. For the hash function, we require the property that $(1, h_{d\text{-wise}}(\mathsf{ID}_1)), \ldots, (1, h_{d\text{-wise}}(\mathsf{ID}_d))$ are linearly independent over $\mathbb{Z}_q$ for mutually distinct $\mathsf{ID}_1, \ldots, \mathsf{ID}_d$. Let us postpone the construction of such a hash function to the main body. Assuming that we have such a hash function, we are now able to prove $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j_1)} \wedge \cdots \wedge \mathsf{E}^{(j_{d-1})}] = q^{-cd}$ by the same linear algebraic discussion

we have done. We can then approximate the value of $\gamma(\vec{\mathsf{ID}})$ within error $Q^{d-1}q^{-cd}$ and this leads to the improved reduction cost of $\gamma_{\mathsf{min}}\epsilon = \Theta(\epsilon^{1+1/(d-1)}/Q)$. Furthermore, by setting $d = \omega(1)$, we can competelty eliminate the dependence of γ_{min} on ϵ to achieve $\gamma_{\mathsf{min}} = O(1/Q)$, which leads to $\gamma_{\mathsf{min}}\epsilon = O(\epsilon/Q)$. Note that the above change in the partitioning function increases the size of the key K, since the output length L_d of $h_{d\text{-}\mathsf{wise}}$ is about $d\ell$, which is d times longer than the input.

2.8 Partitioning Based on Substring Matching

Here, we demonstrate that our technique can lead to tighter analysis also for the partitioning based on the substring matching [Lys02], which has been a useful tool in constructing adaptively secure IBEs and VRFs [BB04b, CHKP10, Yam17, Kat17, Bit17, Koh19]. Here, we focus on the application to IBE, though our analysis is applicable to VRF as well, as is done in Sec. 7. To describe the partitioning function, we introduce an error correcting code $\mathsf{Encode} : \{0,1\}^{\ell} \to \{0,1\}^{n}$ with relative distance $0 < c < 1/2$ and output length n.[9] Then, the identity space $\{0,1\}^{\ell}$ is partitioned as follows:

$$\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{ID}) = \begin{cases} 0 & \text{if} \quad \sigma_i = \mathsf{Encode}(\mathsf{ID})_{I_i} \quad \forall i \in [\eta] \\ 1 & \text{otherwise} \end{cases}, \tag{8}$$

where the secret information K is in the form of $K = \{(I_i, \sigma_i)\}_{i\in[\eta]}$ and $I_i \in [n]$ and $\sigma_i \in \Sigma$ for all $i \in [n]$, with η being a parameter that will be chosen later. We choose $I = (I_i)_{i\in[\eta]}$ so that I constitutes a random subset of $[n]$ and $\sigma_i \xleftarrow{\$} \{0,1\}$ for each i.

Bellare-Ristenpart-style analysis. We then analyze $\gamma(\vec{\mathsf{ID}})$. Let us start with a Bellare-Ristenpart-style analysis. Denoting the event that $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{ID}^*) = 0$ holds by E^* and the event that $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{ID}^{(j)}) = 0$ holds for $j \in [Q]$ by $\mathsf{E}^{(j)}$, Eq. (4) can be shown to hold by the same analysis as in Sec. 2.4. We then proceed to bound the error term $\sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$. Noting that it is straightforward to see $\Pr[\mathsf{E}^*] = 2^{-\eta}$, we evaluate $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$:

$$\begin{aligned}
\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}] &= \Pr\left[(\mathsf{Encode}(\mathsf{ID}^*)_{I_i} = \sigma_i \ \forall i \in [\eta]) \wedge I \subseteq \underbrace{\{k : \mathsf{Encode}(\mathsf{ID}^*)_k = \mathsf{Encode}(\mathsf{ID}^{(j)})_k\}}_{:=J}\right] \\
&= \Pr\left[\left(\mathsf{Encode}(\mathsf{ID}^*)_{I_i} = \sigma_i \ \forall i \in [\eta']\right) \;\middle|\; I \subseteq J\right] \cdot \Pr[I \subseteq J\}] \\
&= 2^{-\eta} \cdot \prod_{i=0}^{\eta-1} \left(\frac{\#J - i}{n - i}\right) \\
&\leq 2^{-\eta} \cdot \prod_{i=0}^{\eta-1} \left(\frac{(1-c)n - i}{n - i}\right) \\
&\leq 2^{-\eta}(1-c)^{\eta}
\end{aligned} \tag{9}$$

where the third equation follows from $\sigma_i \xleftarrow{\$} \{0,1\}$ and by the fact that I is a random subset of $[n]$ and the first inequality follows from the fact that the relative distance of Encode is c, which in turn implies $J \leq (1-c)n$. From the above analysis, we can see that the error term in Eq. (4) can be bounded by $Q \cdot 2^{-\eta}(1-c)^{\eta}$. It remains to choose γ_{min} and η so that $\gamma_{\mathsf{min}} \leq 2^{-\eta} - Q2^{-\eta}(1-c)^{\eta}$ and $Q \cdot 2^{-\eta}(1-c)^{\eta} \leq \gamma_{\mathsf{min}}\epsilon/3$ hold. Combining these inequalities, we have $Q \cdot 2^{-\eta}(1-c)^{\eta} \leq 2^{-\eta}\epsilon/3$. To

[9] Many previous works (e.g., [BB04b, CHKP10]) primarily focus on the encoding function and call it "admissible hash". In this paper, we use the term partitioning based on substring matching following [Bit17, Koh19].

satisfy this, we choose $\eta = \log_{1/1-c}(3Q/\epsilon)$, which leads to the reduction cost $\gamma_{\mathsf{min}}\epsilon = \Theta(\epsilon^{1+\mu}/Q^{\mu})$, where $\mu = 1/(\log 1/(1-c))$. Note that we have $\mu > 1$ and by approaching c to $1/2$, it is possible to make μ approach to 1 as closely as one wants. The security proofs for many IBE and VRF scehems [Jag15, Yam17, Kat17, Koh19] essentially depend on the above analysis and derive the above reduction cost.

Our improved analysis. We then move to explain our finer-grained analysis. Now, by the same analysis as Sec. 2.5 using Bonferroni's inequality, we can derive Eq. (5). We then set $\widetilde{\gamma}(\vec{\mathsf{ID}}) := \Pr[\mathsf{E}^*] - \sum_{j\in[Q]} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$. Similarly to Sec. 2.7, it is straightforward to compute $\widetilde{\gamma}(\vec{\mathsf{ID}})$ efficiently, since we can use Eq. (9) to compute each of $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)}]$. We then bound the error term $\sum_{j,k} \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}]$ in Eq. (5). For doing that, we need an extra property for Encode that we call the *small triple overlap property*. Namely, we need for an arbitrary but mutually distinct $\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3 \in \{0,1\}^{\ell}$ to satisfy,

$$\#\left\{\iota \in [n] : \mathsf{Encode}(\mathsf{x}_1)_\iota = \mathsf{Encode}(\mathsf{x}_2)_\iota = \mathsf{Encode}(\mathsf{x}_3)_\iota\right\} \leq (1-c)^2 n.$$

We defer the construction of such code to the end of this subsection and continue the analysis. We now bound each of $\Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}]$:

$$\begin{aligned}
& \Pr[\mathsf{E}^* \wedge \mathsf{E}^{(j)} \wedge \mathsf{E}^{(k)}] \\
= \; & \Pr\left[(\mathsf{Encode}(\mathsf{ID}^*)_{I_i} = \sigma_i \; \forall i \in [\eta]) \wedge I \subseteq \underbrace{\{\iota : \mathsf{Encode}(\mathsf{ID}^*)_\iota = \mathsf{Encode}(\mathsf{ID}^{(j)})_\iota = \mathsf{Encode}(\mathsf{ID}^{(k)})_\iota\}}_{:=L}\right] \\
= \; & \Pr\left[(\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i \; \forall i \in [\eta]) \;\middle|\; I \subseteq L\right] \cdot \Pr[I \subseteq L] \\
= \; & 2^{-\eta} \cdot \prod_{i=0}^{\eta-1} \left(\frac{\#L - i}{n-i}\right) \\
\leq \; & 2^{-\eta} \cdot \prod_{i=0}^{\eta-1} \left(\frac{(1-c)^2 n - i}{n-i}\right) \\
\leq \; & 2^{-\eta}(1-c)^{2\eta}.
\end{aligned}$$

where we use the small triple overlap property in the first inequality. From the above analysis, we can see that the error term in Eq. (5) can be bounded by $Q^2 2^{-\eta}(1-c)^{2\eta}$. It remains to choose γ_{min} and η so that $\gamma_{\mathsf{min}} \leq 2^{-\eta} - Q \cdot 2^{-\eta}(1-c)^{\eta}$ and $Q^2 \cdot 2^{-\eta}(1-c)^{2\eta} \leq \gamma_{\mathsf{min}}\epsilon/3$ hold. From the both inequalities, we can derive $Q^2(1-c)^{2\eta} \leq \epsilon/3$. We then take $\eta = \log_{1/1-c}(3Q/\sqrt{\epsilon})$, which leads to the reduction cost $\gamma_{\mathsf{min}}\epsilon = \Theta(\epsilon^{1+\mu/2}/Q^{\mu})$, where $\mu = \log 1/(1-c)$. Note that we can make μ approach 1 as closely as one wants by approaching c to $1/2$. This improves the reduction cost of previous works $\gamma_{\mathsf{min}}\epsilon = \Theta(\epsilon^{1+\mu}/Q^{\mu})$ by a factor of $\epsilon^{\mu/2}$. Again, the reason why the improvement is possible is that we use the finer-grained approximation of $\gamma(\vec{\mathsf{ID}})$ using Bonferroni's inequality to represent the error terms as a higher order polynomial of $(1-c)$. This allows us to take η smaller, which leads to better advantage.

Instantiating Encode. We then discuss how to instantiate Encode. Unfortunately, we do not know explicit constructions of a function with the small triple overlap property, where an explicit construction refers to a deterministic algorithm that takes as input n, ℓ, and ID and outputs $\mathsf{Encode}(\mathsf{ID})$. Instead, we observe that a randomly chosen 3-wise independent hash function satisfies this property with overwhelming probability under specific parameter settings. Therefore, in applications to IBEs/VRFs, we choose a random 3-wise independent hash function and append

it to the public parameters as the description of Encode. The description size of Encode is much shorter than any other part of the parameters in our application and does not harm the efficiency of the construction. In addition, we observe that this does not harm the security of the constructions either. We defer to the details to the main body.

Polynomial-size alphabet variant. Finally, we discuss the variant of the function $\mathsf{F}_{\mathsf{SSM}}$ with polynomial-size alphabets, where the underlying encoding function has codewords with a polynomial-size alphabet. Namely, we have $\mathsf{Encode} : \{0,1\}^{\ell} \to \Sigma^n$ for a polynomial size Σ, rather than $\Sigma = \{0,1\}$. While many previous works primarily focused on binary encoding functions when constructing IBEs/VRFs [BB04b, CHKP10, Yam17, Kat17], Kohl [Koh19] showed that using encoding functions with a polynomial-size alphabet can be useful when constructing VRF schemes with a compact proof size. While she uses Reed-Solomon code, we replace it with a 3-wise independent hash function. Since a 3-wise independent hash function achieves larger relative distance c than the Reed-Solomon code (w.h.p), using it is quite beneficial. We can improve the overall parameter size of her construction even if we have to add the description of Encode to the public parameter as the description size is small. Furthermore, we can also improve the reduction cost because of the larger relative distance of the 3-wise independent hash function. On top of the improvement described above, we can further apply our finer-grained analysis to the variant with polynomial-size alphabet, since the underlying encoding function satisfies the small triple overlap property. This leads to a tighter analysis that achieves $\gamma_{\mathsf{min}}\epsilon = O(\epsilon^{1.5}/Q)$.

2.9 Overview for Our Construction of VRF

Finally, we present an overview of our construction of VRF, which is built upon Kohl's construction [Koh19]. As mentioned in Sec. 2.8, by merely substituting the underlying error-correcting code in her construction with ours, we can improve both efficiency and reduction cost. We further reduce the public parameter (i.e., verification key) size to be sublinear by modifying the algebraic structure of the scheme.

Let us first start with the high level overview of the construction by Hofheinz and Jager [HJ16]. Informally speaking, they reduce the problem of constructing VRF to the construction of a function $\mathbf{v}(\cdot)$ that maps a VRF input x to a vector in $\mathbb{Z}_p^k$ with the following properties: First, $g^{\mathbf{v}(\mathsf{x})}$ can be certified by the public parameter with the help of a proof, where g is a generator of a pairing group. Furthermore, $\mathbf{v}(\cdot)$ should be compatible with the partitioning $\mathsf{F}_{\mathsf{SSM}}$ (defined in Eq. (8)) in the security proof. Namely, we require that if x is in the controlled set, then $\mathbf{v}(\mathsf{x})$ is in certain subspace $\mathcal{U}$ of $\mathbb{Z}_p^k$ with dimension $k-1$ and otherwise it is outside of $\mathcal{U}$. In the subsequent work, Kohl [Koh19] follows the framework but instantiates $\mathbf{v}(\cdot)$ in a new way, resulting in the improvement on the proof size of the construction by Hofheinz and Jager. In this overview, we present her construction of $\mathbf{v}(\cdot)$ associated with $\mathsf{F}_{\mathsf{SSM}}$ in the oversimplified setting, where we set $\eta = 1$ and Encode to be an identity map. In this setting, the secret information K consists of a pair of an index and a bit $(i^*, \sigma^*) \in [\ell] \times \{0,1\}$. We have $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$ if and only if the i^*-th bit x_{i^*} of x equals to σ^*. In her construction, she defines

$$\mathbf{v}(\mathsf{x}) = \sum_{i \in [\ell]} \mathbf{L}_{i,\mathsf{x}_i}^{\top} \mathbf{u},$$

where $\mathbf{u} \in \mathbb{Z}_p^k$, and $\{\mathbf{L}_{i,b} \in \mathbb{Z}_p^{k \times k}\}_{i \in [n], b \in \{0,1\}}$ are parameters fixed in the system. Here, $\mathbf{u}$ is chosen uniformly at random and $\mathbf{L}_{i,b}^{\top}$ is chosen so that its image is contained in $\mathcal{U}$ if $(i,b) \neq (i^*, \sigma^*)$ and it is full-rank when $(i,b) = (i^*, \sigma^*)$. We have that over the random choice of $\mathbf{u}$, $\mathbf{v}(\mathsf{x})$ is in $\mathcal{U}$ if and only if $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$ with high probability as desired. This can be observed from the fact that

when $(i,b) \neq (i^*,b^*)$, $\mathbf{L}_{i,b}^\top \mathbf{u}$ is within $\mathcal{U}$; otherwise, $\mathbf{L}_{i,b}^\top \mathbf{u}$ is distributed uniformly at random across $\mathbb{Z}_p^k$ and has a negligible probability of falling into the subspace $\mathcal{U}$. When converting the above function into a VRF, we must incorporate all $g^{\mathbf{u}}$ and $\{g^{\mathbf{L}_{i,b}}\}_{i\in[\ell],b\in\{0,1\}}$ in the public parameter. This results in $O(\ell)$ group elements, which is rather large.

To reduce the public parameter, we indirectly define $\{\mathbf{L}_{i,b}\}_{i,b}$ by the combination of smaller number of matrices $\{\mathbf{M}_j\}_{j\in[\ell_1]}$ and $\{\mathbf{N}_k\}_{k\in[\ell_2]}$ as

$$\mathbf{L}_{i,b} = \mathbf{M}_{S_1(i,b)}\mathbf{N}_{S_2(i,b)},$$

and publish $\{g^{\mathbf{M}_j}\}_{j\in[\ell_1]}$ and $\{g^{\mathbf{N}_k}\}_{k\in[\ell_2]}$ instead of $\{g^{\mathbf{L}_{i,b}}\}_{i,b}$. Here, $S_1 : [\ell]\times\{0,1\} \to [\ell_1]$ and $S_2 : [\ell]\times\{0,1\} \to [\ell_2]$ are arbitrary efficiently computable maps such that $(i,b) \mapsto (S_1(i,b), S_2(i,b))$ is injective. To be able to define such a map, it suffices to set $\ell_1 = \ell_2 = \lceil\sqrt{2\ell}\rceil$. This reduces the number of group elements in the verification key to be $O(\ell_1+\ell_2) = O(\sqrt{\ell})$ from $O(\ell)$.

We then show that we can make the function compatible with $\mathsf{F}_{\mathsf{SSM}}$ by appropriately defining the matrices. Let $j^* = S_1(i^*,\sigma^*)$ and $k^* = S_2(i^*,\sigma^*)$. We then set the matrix $\mathbf{M}_j^\top$ (resp., $\mathbf{N}_k^\top$) so that its image is in $\mathcal{V}$ (resp., $\mathcal{U}$) if $j \neq j^*$ (resp., $k \neq k^*$), where $\mathcal{V}$ is some subspace of $\mathbb{Z}_p^k$ with dimension $k-1$. Furthermore, we set $\mathbf{M}_{j^*}$ and $\mathbf{N}_{k^*}$ to be full-rank matrices with the constraint that $\mathbf{N}_{k^*}^\top$ maps a vector in $\mathcal{V}$ to a vector in $\mathcal{U}$. We have that with high probability over the choice of $\mathbf{u}$, $\mathbf{L}_{i,b}^\top\mathbf{u} = \mathbf{N}_{S_2(i,b)}^\top\mathbf{M}_{S_1(i,b)}^\top\mathbf{u}$ is in $\mathcal{U}$ if and only if $(S_1(i,b),S_2(i,b)) = (j^*,k^*)$, which is equivalent to $(i,b) = (i^*,\sigma^*)$. This can be seen by the case analysis. If $S_2(i,b) \neq k^*$, $\mathbf{N}_{S_2(i,b)}^\top\mathbf{M}_{S_1(i,b)}^\top\mathbf{u}$ is in $\mathcal{U}$. Otherwise, there are two cases to consider: If $S_2(i,b) = k^*$ and $S_1(i,b) \neq j^*$, we have that $\mathbf{M}_{S_1(i,b)}^\top\mathbf{u}$ is in $\mathcal{V}$. This implies $\mathbf{L}_{i,b}^\top\mathbf{u}$ is in $\mathcal{U}$, since $\mathbf{N}_{S_2(i,b)}^\top$ maps an element in $\mathcal{V}$ to $\mathcal{U}$. If $S_2(i,b) = k^*$ and $S_1(i,b) = j^*$, both $\mathbf{N}_{S_2(i,b)}^\top$ and $\mathbf{M}_{S_1(i,b)}^\top$ are full-rank, which implies that $\mathbf{L}_{i,b}^\top\mathbf{u}$ is distributed uniformly at random over $\mathbb{Z}_p^k$, meaning that the vector falls into $\mathcal{U}$ only with negligible probability. The above observation immediately implies that $\mathbf{v}(\mathsf{x})$ is in $\mathcal{U}$ if and only if $\mathsf{F}_{\mathsf{SSM}}(K,\mathsf{x}) = 1$ as desired.

One may ask why we limit ourselves to only two sequences of matrices (i.e., $\{\mathbf{M}_j\}_j$ and $\{\mathbf{N}_k\}_k$). Namely, by considering three sequences, we could potentially achieve a further reduction in the size of the verification key to $O(\ell^{1/3})$. The answer is that because we do not know how to give a short proof to certify $g^{\mathbf{v}(x)}$. In the above example, publishing $\pi = g^{\sum_i \mathbf{L}_{i,\mathsf{x}_i}}$ suffices to certify $g^{\mathbf{v}(\mathsf{x})}$: We can verify the value of $g^{\mathbf{v}(\mathsf{x})}$ by checking $e(g^{\mathbf{v}(\mathsf{x})},g) \stackrel{?}{=} e(\pi^\top, g^{\mathbf{u}})$ and $e(\pi,g) \stackrel{?}{=} \sum_{i,b} e(g^{\mathbf{M}_{S_1(i,b)}}, g^{\mathbf{N}_{S_2(i,b)}})$. Importantly, by the pairing, we can easily check the quadratic forms on the exponent. However, we cannot do this for the cubic form. We leave the problem of further reducing the size of the verification key while maintaining the short proof as an open problem.

3 Preliminaries

3.1 Notations

For a distribution $\mathcal{D}$, $x \in \mathcal{D}$ means $\Pr[y = x : y \xleftarrow{\$} \mathcal{D}] > 0$. With an abuse of notations, we extend this to a set of distributions, that is, $x \in \mathcal{D}\cup\mathcal{D}'$ means $\Pr[y = x \vee y' = x : y \xleftarrow{\$} \mathcal{D}, y \xleftarrow{\$} \mathcal{D}'] > 0$ For an algorithm A, $\mathsf{A}(x)$ denotes the output distribution of A on input x.

Definition 1 (Hard Decision Problem)**.** *We say a family of pairs of distributions $\mathcal{D} := \{(\mathcal{D}_{\lambda,0},\mathcal{D}_{\lambda,1})\}_\lambda$ is a hard decision problem if the following advantage is negligible for all PPT adversary A.*

$$\mathsf{Adv}^{\mathcal{D}}(\mathsf{A}) := \left|\Pr[\mathsf{A}(1^\lambda,\psi) = 1 : \psi \leftarrow \mathcal{D}_0] - \Pr[\mathsf{A}(1^\lambda,\psi) = 1 : \psi \leftarrow \mathcal{D}_1]\right|.$$

3.2 Identity-based Encryption

We provide the definition of an identity-based encryption (IBE) scheme.

Definition 2 (Identity-based Encryption)**.** *An* identity-based encryption (IBE) *scheme with an (efficiently sampleable) message space $\mathcal{M}$ and identity space $\{0,1\}^\ell$ is defined by the following four algorithms.*

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

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$**:** *It takes as input a master public key* mpk*, a master secret key* msk*, and an identity $\mathsf{ID} \in \{0,1\}^\ell$ and outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$.*

$\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M}) \to \mathsf{ct}$**:** *It takes as input a master public key* mpk*, an identity $\mathsf{ID} \in \{0,1\}^\ell$, and a message* M *and outputs a ciphertext* ct*.*

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to \mathsf{M}$ or $\perp$**:** *It takes as input a master public key* mpk*, a private key $\mathsf{sk}_{\mathsf{ID}}$, and a ciphertext* ct *and outputs the message* M *or* $\perp$*.*

An IBE scheme satisfies correctness and IND-CPA security, defined below.

Definition 3 (Correctness)**.** *An* IBE *scheme is* correct *if for all $\lambda \in \mathbb{N}$, all $\mathsf{ID} \in \{0,1\}^\ell$, and all $\mathsf{M} \in \mathcal{M}$, the following holds*

$$\Pr\left[\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) = \mathsf{M} : \begin{array}{r}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})\end{array}\right] = 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness of the algorithms.

Definition 4 (IND-CPA Security)**.** *To define* IND-CPA *security of an* IBE *scheme, we consider the following game between a challenger and an adversary* A*.*

Setup. *The challenger generates* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives* mpk *to* A*.*

Phase 1. A *can adaptively make key-extraction queries. When* A *submits $\mathsf{ID} \in \{0,1\}^\ell$, the challenge generates $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$ and returns $\mathsf{sk}_{\mathsf{ID}}$ to* A*.*

Challenge. *At any point,* A *can make a challenge query by submitting a messages $\mathsf{M}_0 \in \mathcal{M}$ and an identity $\mathsf{ID}^* \in \{0,1\}^\ell$, never queried in Phase 1. The challenger picks a random coin $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. If $\mathsf{coin} = 0$, it generates $\mathsf{ct}_0^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}_0)$. If $\mathsf{coin} = 1$, it samples a random message $\mathsf{M}_1 \leftarrow \mathcal{M}$, generates $\mathsf{ct}_1^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}_1)$, and returns $\mathsf{ct}_{\mathsf{coin}}^*$ to* A*.*

Phase 2 A *can continue making key-extraction queries with the added restriction that it can only query $\mathsf{ID} \in \{0,1\}^\ell$ such that $\mathsf{ID} \neq \mathsf{ID}^*$.*

Guess. *Finally,* A *outputs a guess $\widehat{\mathsf{coin}}$ for* coin*.*

The advantage of A *is defined as $\mathsf{Adv}_{\mathsf{IBE}}^{\mathsf{IND\text{-}CPA}}(\mathsf{A}) = |\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - 1/2|$. We say that an adversary* A *is a (t, Q, ϵ)-adversary if* A *runs in time t, makes Q key-extraction queries, and has advantage $\mathsf{Adv}_{\mathsf{IBE}}^{\mathsf{IND\text{-}CPA}}(\mathsf{A}) \geq \epsilon$. We say that an* IBE *scheme is* (t, Q, ϵ)-random-or-challenge-plaintext-attack secure *if there is no (t, Q, ϵ)-adversary.*

Note the above definition is identical, up to a constant factor 2, to the alternative notion of IND-CPA security where the adversary submits two messages $(\mathsf{M}_0, \mathsf{M}_1)$ of its choice.

3.3 Verifiable Random Function

We provide the definition of a verifiable random function (VRF) scheme.

Definition 5 (Verifiable Random Function)**.** *A* verifiable random function (VRF) *with (efficiently sampleable) input and output spaces* $(\{0,1\}^\ell, \mathcal{Y})$ *is defined by the following three algorithms.*

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

$\mathsf{Eval}(\mathsf{sk}, \mathsf{x}) \to (\mathsf{y}, \pi)$**:** *It takes as input a secret key* sk *and an input* $\mathsf{x} \in \{0,1\}^\ell$ *and outputs a value* $\mathsf{y} \in \mathcal{Y}$ *and proof* π*.*

$\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \mathsf{y}, \pi) \to 1$ or 0**:** *It takes as input a verification key* vk*, input* $\mathsf{x} \in \{0,1\}^\ell$*,* $\mathsf{y} \in \mathcal{Y}$*, and a proof* π *and outputs a bit.*

A VRF satisfies correctness, unique provability, and pseudorandomness, defined below.

Definition 6 (Correctness)**.** *For all* $\lambda \in \mathbb{N}$*,* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*,* $\mathsf{x} \in \{0,1\}^\ell$*, and* $(\mathsf{y}, \pi) \leftarrow \mathsf{Eval}(\mathsf{sk}, \mathsf{x})$*, we have* $\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \mathsf{y}, \pi) = 1$*.*

Definition 7 (Unique Provability)**.** *For all* $\mathsf{vk} \in \{0,1\}^*$ *(not necessarily generated by* Gen*) and all* $\mathsf{x} \in \{0,1\}^\ell$*, there does not exist* $(\mathsf{y}_0, \pi_0, \mathsf{y}_1, \pi_1)$ *such that* $\mathsf{y}_0 \neq \mathsf{y}_1$ *and* $\mathsf{Verify}(\mathsf{vk}, \mathsf{vk}, \mathsf{x}, \mathsf{y}_0, \pi_0) = \mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \mathsf{y}_1, \pi_1) = 1$*.*

Definition 8 (Pseudorandomness)**.** *To define pseudorandomness of a* VRF*, we consider the following game between a challenger and an adversary* A*.*

Setup. *The challenger generates* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and gives* vk *to* A*.*

Phase 1. A *can adaptively make evaluation queries. When* A *submits* $\mathsf{x} \in \{0,1\}^\ell$*, the challenger generates* $(\mathsf{y}, \pi) \leftarrow \mathsf{Eval}(\mathsf{sk}, \mathsf{x})$ *and returns* (y, π) *to* A*.*

Challenge. *At any point,* A *can make a challenge query by submitting* x^**, never queried in Phase 1. The challenger picks a random coin* $\mathsf{coin} \xleftarrow{\$} \{0,1\}$*. If* $\mathsf{coin} = 0$*, it generates* $(\mathsf{y}_0^*, \pi_0^*) \leftarrow \mathsf{Eval}(\mathsf{sk}, \mathsf{x}^*)$ *. If* $\mathsf{coin} = 1$*, it picks* $\mathsf{y}_1^* \leftarrow \mathcal{Y}$*. It returns* $\mathsf{y}_{\mathsf{coin}}^*$ *to* A*.*

Phase 2 A *can continue making evaluation queries with the added restriction that it can only query* $\mathsf{x} \in \{0,1\}^\ell$ *such that* $\mathsf{x} \neq \mathsf{x}^*$*.*

Guess. *Finally,* A *outputs a guess* $\widehat{\mathsf{coin}}$ *for* coin*.*

The advantage of A *is defined as* $\mathsf{Adv}^{\mathsf{rand}}_{\mathsf{VRF}}(\mathsf{A}) = |\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - 1/2|$*. We say that an adversary* A *is a* (t, Q, ϵ)*-adversary if* A *runs in time* t*, makes* Q *evaluation queries, and has advantage* $\mathsf{Adv}^{\mathsf{rand}}_{\mathsf{VRF}}(\mathsf{A}) \geq \epsilon$*. We say that the* VRF *is* (t, Q, ϵ)*-pseudorandom if there is no* (t, Q, ϵ)*-adversary.*

3.4 Bonferroni's Inequality

We will use the Bonferroni's inequality [Bon36], which is a generalization of the union bound. The inequality is obtained by cutting the tail of inclusion-exclusion principle.

Lemma 1. *Let* $\mathsf{E}_1, \ldots, \mathsf{E}_n$ *be events in a probability space. Then, the following inequalities hold.*

$$\Pr\left[\bigvee_{i=1}^{n} \mathsf{E}_i\right] \leq \sum_{j=1}^{k} (-1)^{j-1} \cdot \sum_{1 \leq \ell_1 < \cdots < \ell_j \leq n} \Pr\left[\bigwedge_{i=1}^{j} \mathsf{E}_{\ell_i}\right] \quad \textit{for any odd } k \in [n],$$

$$\Pr\left[\bigvee_{i=1}^{n} \mathsf{E}_i\right] \geq \sum_{j=1}^{k} (-1)^{j-1} \cdot \sum_{1 \leq \ell_1 < \cdots < \ell_j \leq n} \Pr\left[\bigwedge_{i=1}^{j} \mathsf{E}_{\ell_i}\right] \quad \textit{for any even } k \in [n].$$

4 A Finer Grained Analysis of the Artificial Abort Paradigm

Our main technical contribution is to provide a more fine grained analysis of Bellare and Ristenpart [BR09] by further relying on the artificial abort paradigm [Wat05]. In this section, we divorce the artificial abort paradigm from security proofs of a particular cryptographic primitive. Instead, we provide a statistical theorem that extracts the essence of the paradigm. Looking ahead, in Sec. 5, we will relate the following statistical theorem to concrete cryptographic primitives using a tool called *partitioning function with approximation*. This allows for a more modular proof of IBE and VRF schemes, as we illustrate in Sec. 6 and 7.

Theorem 1. *Let* $\mathcal{T}$ *be a finite set named the* transcript *space. Let* $\mathcal{D} : \{0,1\} \times \{0,1\} \times \mathcal{T} \to [0,1]$ *be an arbitrary distribution. Let* $\gamma_{\mathsf{min}} > 0$ *be a positive real and* $\gamma : \mathcal{T} \to [0,1]$ *and* $\widetilde{\gamma} : \mathcal{T} \to [0,1]$ *be functions such that* $\gamma(\mathsf{T}) \geq \widetilde{\gamma}(\mathsf{T}) \geq \gamma_{\mathsf{min}}$ *for all* transcipts $\mathsf{T} \in \mathcal{T}$.

Consider a distribution $\mathcal{D}^* : \{0,1\} \times \{0,1\} \times \mathcal{T}$ *defined through the following procedure:*

1. *Sample* $(\mathsf{coin}, \widehat{\mathsf{coin}}, \mathsf{T}) \xleftarrow{\$} \mathcal{D}$.
2. *With probability* $\gamma(\mathsf{T})$, *set* $\mathsf{coin}' \leftarrow \widehat{\mathsf{coin}}$ *and with probability* $1 - \gamma(\mathsf{T})$, *sample a uniformly random* $\mathsf{coin}' \xleftarrow{\$} \{0,1\}$. *The later event is called* Bad. *If* $\neg\mathsf{Bad}$, *it further executes Item 3.*
3. *With probability* $1 - \gamma_{\mathsf{min}}/\widetilde{\gamma}(\mathsf{T})$, *replace* coin' *with a uniformly random* $\mathsf{coin}' \xleftarrow{\$} \{0,1\}$. *This event is called* AAbort, *short for* artificial abort.
4. *Output* $(\mathsf{coin}, \mathsf{coin}', \mathsf{T})$.

Lastly, define

$$\epsilon = \left| \Pr_{(\mathsf{coin}, \widehat{\mathsf{coin}}, \mathsf{T}) \xleftarrow{\$} \mathcal{D}} \left[\widehat{\mathsf{coin}} = \mathsf{coin}\right] - \frac{1}{2} \right| \quad \textit{and} \quad \epsilon^* = \left| \Pr_{(\mathsf{coin}, \mathsf{coin}', \mathsf{T}) \xleftarrow{\$} \mathcal{D}^*} \left[\mathsf{coin}' = \mathsf{coin}\right] - \frac{1}{2} \right|.$$

Then, if $|\gamma(\mathsf{T}) - \widetilde{\gamma}(\mathsf{T})| < \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon$ *holds for all transcripts* $\mathsf{T} \in \mathcal{T}$, *we have* $\epsilon^* > \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon$.

Before providing the proof, we explain some intuition of the theorem. In the context of security proofs, coin denotes the random challenge bit sampled by the challenger and $\widehat{\mathsf{coin}}$ denotes the guess output by the adversary A. The advantage of A is thus ϵ. Bad denotes the typical event that the reduction fails. For example, in the context of IBE schemes, Bad can denote the event that the reduction cannot answer the key-extraction query or cannot simulate the challenge ciphertext. In such a case, since the reduction cannot properly simulate the game for A, it will output a random coin' as A's output. AAbort is the more interesting event. In this case, while the reduction is able to simulate A till the end of the game and obtains $\widehat{\mathsf{coin}}$, it will ignore this and output a random coin' with some probability. The term *artificial* abort stems from the fact that the reduction is

ignoring A's output even if it might be the case $\mathsf{coin} = \widehat{\mathsf{coin}}$. While counter intuitive, The artificial abort paradigm states that the reduction's advantage can degrade by at most a factor $\gamma_{\mathsf{min}}/3$. In other words, the quality of the reduction is dictated by how large γ_{min} can be; the larger the γ_{min}, the better the reduction is.

We now present the proof of Theorem 1.

Proof of Theorem 1. For $\mathsf{T} \in \mathcal{T}$, let $\mathsf{E}(\mathsf{T})$ be the event that T is sampled by $\mathcal{D}$. Note that T is sampled by $\mathcal{D}^*$ with the same probability as by $\mathcal{D}$. Then, we have the following, where unless stated otherwise, we assume the probability is taken over the randomness of sampling from $\mathcal{D}^*$:

$$
\begin{aligned}
\epsilon^* &= \left| \Pr\left[\mathsf{coin}' = \mathsf{coin}\right] - \frac{1}{2} \right| \\
&= \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{coin}' = \mathsf{coin} \wedge \mathsf{E}(\mathsf{T})\right] - \frac{1}{2} \right| && (10) \\
&= \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \left(\Pr\left[\mathsf{coin}' = \mathsf{coin} \wedge \neg\mathsf{Bad} \middle| \mathsf{E}(\mathsf{T})\right] + \Pr\left[\mathsf{coin}' = \mathsf{coin} \wedge \mathsf{Bad} \middle| \mathsf{E}(\mathsf{T})\right] - \frac{1}{2} \right) \right| && (11) \\
&= \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \left(\gamma(\mathsf{T}) \cdot \Pr\left[\mathsf{coin}' = \mathsf{coin} \middle| \mathsf{E}(\mathsf{T}) \wedge \neg\mathsf{Bad}\right] + \frac{1}{2} \cdot (1 - \gamma(\mathsf{T})) - \frac{1}{2} \right) \right| && (12) \\
&= \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \gamma(\mathsf{T}) \left(\Pr\left[\mathsf{coin}' = \mathsf{coin} \wedge \neg\mathsf{AAbort} \middle| \mathsf{E}(\mathsf{T}) \wedge \neg\mathsf{Bad}\right] \right.\right. \\
&\qquad \left.\left. + \Pr\left[\mathsf{coin}' = \mathsf{coin} \wedge \mathsf{AAbort} \middle| \mathsf{E}(\mathsf{T}) \wedge \neg\mathsf{Bad}\right] - \frac{1}{2} \right) \right| && (13) \\
&= \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \gamma(\mathsf{T}) \left(\frac{\gamma_{\mathsf{min}}}{\widetilde{\gamma}(\mathsf{T})} \cdot \Pr\left[\mathsf{coin}' = \mathsf{coin} \middle| \mathsf{E}(\mathsf{T}) \wedge \neg\mathsf{Bad} \wedge \neg\mathsf{AAbort}\right] + \frac{1}{2} \cdot \left(1 - \frac{\gamma_{\mathsf{min}}}{\widetilde{\gamma}(\mathsf{T})}\right) - \frac{1}{2} \right) \right| && (14) \\
&= \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \gamma(\mathsf{T}) \frac{\gamma_{\mathsf{min}}}{\widetilde{\gamma}(\mathsf{T})} \left(\Pr_{(\mathsf{coin},\widehat{\mathsf{coin}},\mathsf{T})\xleftarrow{\$}\mathcal{D}} \left[\widehat{\mathsf{coin}} = \mathsf{coin} \middle| \mathsf{E}(\mathsf{T})\right] - \frac{1}{2} \right) \right| && (15) \\
&= \gamma_{\mathsf{min}} \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} \cdot \epsilon(\mathsf{T}) \right| && (16) \\
&= \gamma_{\mathsf{min}} \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \cdot \epsilon(\mathsf{T}) + \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \left(\frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} - 1 \right) \cdot \epsilon(\mathsf{T}) \right| && (17) \\
&\geq \gamma_{\mathsf{min}} \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \cdot \epsilon(\mathsf{T}) \right| - \gamma_{\mathsf{min}} \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \left(\frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} - 1 \right) \cdot \epsilon(\mathsf{T}) \right| && (18) \\
&\geq \gamma_{\mathsf{min}} \left| \sum_{\mathsf{T}\in\mathcal{T}} \Pr\left[\mathsf{E}(\mathsf{T})\right] \cdot \epsilon(\mathsf{T}) \right| - \gamma_{\mathsf{min}} \sum_{\mathsf{T}\in\mathcal{T}} \left| \Pr\left[\mathsf{E}(\mathsf{T})\right] \left(\frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} - 1 \right) \cdot \epsilon(\mathsf{T}) \right| && (19) \\
&= \gamma_{\mathsf{min}} \cdot \epsilon - \gamma_{\mathsf{min}} \sum_{\mathsf{T}\in\mathcal{T}} \left| \frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} - 1 \right| \cdot \left| \Pr\left[\mathsf{E}(\mathsf{T})\right] \cdot \epsilon(\mathsf{T}) \right| && (20)
\end{aligned}
$$

where $\epsilon(\mathsf{T}) = \Pr_{(\mathsf{coin},\widehat{\mathsf{coin}},\mathsf{T})\xleftarrow{\$}\mathcal{D}} \left[\widehat{\mathsf{coin}} = \mathsf{coin} \middle| \mathsf{E}(\mathsf{T})\right] - \frac{1}{2}$. In the above, Eq. (10) and Eq. (11) follow by

the law of total probability, Eq. (12) follows from $\Pr[\mathsf{Bad}|\mathsf{E}(\mathsf{T})] = 1-\gamma(\mathsf{T})$, Eq. (13) follows by the law of total probability, Eq. (14) follows from $\Pr[\mathsf{AAbort}|\mathsf{E}(\mathsf{T}) \wedge \neg\mathsf{Bad}] = 1 - \gamma_{\min}/\widetilde{\gamma}(\mathsf{T})$, Eq. (15) follows from the fact that $\widehat{\mathsf{coin}} = \mathsf{coin}'$ holds conditioned on $\neg\mathsf{BadID} \wedge \neg\mathsf{AAbort}$, Eq. (16) and Eq. (17) are only a change in expression, Eq. (18) and Eq. (19) follows by the triangle inequality, and Eq. (20) follows from the facts that $\left|\sum_{\mathsf{T}\in\mathcal{T}} \Pr[\mathsf{E}(\mathsf{T})]\,\epsilon(\mathsf{T})\right| = \epsilon$ and $|a\cdot b| = |a|\cdot|b|$ for any real numbers a and b.

From our assumption that $|\gamma(\mathsf{T}) - \widetilde{\gamma}(\mathsf{T})| < \gamma_{\mathsf{min}}\epsilon/3$ and $0 < \gamma_{\mathsf{min}} \leq \widetilde{\gamma}(\mathsf{T})$ for all $\mathsf{T} \in \mathcal{T}$, we have

$$
\begin{aligned}
& -\frac{\gamma_{\mathsf{min}}}{3}\cdot\epsilon < \gamma(\mathsf{T}) - \widetilde{\gamma}(\mathsf{T}) < \frac{\gamma_{\mathsf{min}}}{3}\cdot\epsilon \\
\Rightarrow\quad & 1 - \frac{\gamma_{\mathsf{min}}}{3\widetilde{\gamma}(\mathsf{T})}\cdot\epsilon < \frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} < 1 + \frac{\gamma_{\mathsf{min}}}{3\widetilde{\gamma}(\mathsf{T})}\cdot\epsilon \\
\Rightarrow\quad & 1 - \frac{\epsilon}{3} < \frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} < 1 + \frac{\epsilon}{3} \\
\Rightarrow\quad & -\frac{\epsilon}{3} < \frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} - 1 < \frac{\epsilon}{3} \\
\Rightarrow\quad & \left|\frac{\gamma(\mathsf{T})}{\widetilde{\gamma}(\mathsf{T})} - 1\right| < \frac{\epsilon}{3}
\end{aligned}
$$

Using this inequality, we can lower bound Eq. (20) by the following:

$$
\begin{aligned}
& \gamma_{\min}\cdot\epsilon - \gamma_{\min}\cdot\frac{\epsilon}{3}\sum_{\mathsf{T}\in\mathcal{T}} |\Pr[\mathsf{E}(\mathsf{T})]\cdot\epsilon(\mathsf{T})| && (21)\\
=& \gamma_{\min}\cdot\epsilon - \gamma_{\min}\cdot\frac{\epsilon}{3}\left(\sum_{\mathsf{T}\in\mathcal{T}\text{ s.t. }\epsilon(\mathsf{T})\geq 0} |\Pr[\mathsf{E}(\mathsf{T})]\cdot\epsilon(\mathsf{T})| + \sum_{\mathsf{T}\in\mathcal{T}\text{ s.t. }\epsilon(\mathsf{T})<0} |\Pr[\mathsf{E}(\mathsf{T})]\cdot\epsilon(\mathsf{T})|\right) && (22)\\
=& \gamma_{\min}\cdot\epsilon - \gamma_{\min}\cdot\frac{\epsilon}{3}\left(\sum_{\mathsf{T}\in\mathcal{T}\text{ s.t. }\epsilon(\mathsf{T})\geq 0} \Pr[\mathsf{E}(\mathsf{T})]\cdot\epsilon(\mathsf{T}) - \sum_{\mathsf{T}\in\mathcal{T}\text{ s.t. }\epsilon(\mathsf{T})<0} \Pr[\mathsf{E}(\mathsf{T})]\cdot\epsilon(\mathsf{T})\right) && (23)\\
\geq& \frac{\gamma_{\mathsf{min}}}{3}\cdot\epsilon. && (24)
\end{aligned}
$$

In the above, Eq. (23) follows from the sign of $\epsilon(\mathsf{T})$ inside the absolute value, and Eq. (24) follows from the facts that $\sum_{\mathsf{T}\in\mathcal{T}\text{ s.t. }\epsilon(\mathsf{T})\geq 0} \Pr[\mathsf{E}(\mathsf{T})]\,\epsilon(\mathsf{T}) \leq 1$ and $\sum_{\mathsf{T}\in\mathcal{T}\text{ s.t. }\epsilon(\mathsf{T})<0} \Pr[\mathsf{E}(\mathsf{T})]\,\epsilon(\mathsf{T}) \geq -1$. Combining the inequalities, we obtain $\epsilon^* > \frac{\gamma_{\mathsf{min}}}{3}\cdot\epsilon$ as desired. □

Remark 1 (Comparison with Prior Work)**.** *As briefly mentioned in the introduction, the proof of Bellare and Ristenpart [BR09] can be seen as a special case of our Theorem 1. Their proof fixes the approximation function $\widetilde{\gamma}(\mathsf{T}) := \gamma_{\mathsf{min}}$ for all $\mathsf{T} \in \mathcal{T}$. Effectively, this is a special class of reduction* without *performing artificial aborts. As we see in the later sections, a tighter security proof is achieved by fine-tuning $\widetilde{\gamma}(\mathsf{T})$ and tactically performing artificial aborts. While we did not chose to do so, we can generalize our Theorem 1 to capture the proof of Waters [Wat05] as well. Recall that in his proof, $\widetilde{\gamma}(\mathsf{T})$ is not a fixed value but rather a probabilistic value defined through the Monte Carlo method. Accordingly, $|\gamma(\mathsf{T}) - \widetilde{\gamma}(\mathsf{T})| < \frac{\gamma_{\mathsf{min}}}{3}\cdot\epsilon$ will only be satisfied with some probability. As we did not obtain new results with this generalization, we intentionally kept our definition simple to only capture [BR09].*

5 Partitioning Function with Approximation

In this section, we introduce a tool called *partitioning function with approximation* allowing us to naturally use the finer grained artificial abort paradigm in Theorem 1 to prove tighter security of a wide class of cryptographic primitives.

5.1 Overview

A partitioning function *without* approximation was first introduced by Yamada [Yam17]. Let us use IBE schemes as a representative example to get a flavor of this tool. A partitioning function allows the reduction to secretly partition the identity space into two sets of exponential size: the reduction can answer key-extraction queries on one set and embed a hard problem into the challenge ciphertext on the other set. The partition is made in a meticulous manner so that there is a noticeable probability that the adversary's key-extraction queries and the challenge identity fall in the correct sets. Looking at Theorem 1, the probability that the partitioning fails (e.g., the reduction cannot answer the key-extraction query) is denoted as Bad, occurring with probability $1-\gamma(\vec{\mathsf{ID}})$, where $\vec{\mathsf{ID}}$ is the sequence of identities queried by the adversary. Most prior works using (explicitly or implicitly) partitioning functions [Yam16, Jag15, KY16, Yam17, Bit17] rely on the analysis of Bellare and Ristenpart [BR09]. They approximate $\gamma(\vec{\mathsf{ID}})$ by the trivial lower bound $\widetilde{\gamma}(\vec{\mathsf{ID}})=\gamma_{\mathsf{min}}$, in which case the probability of an artificial abort AAbort occurring becomes $1-\gamma_{\mathsf{min}}/\widetilde{\gamma}(\vec{\mathsf{ID}})=0$. Consequently, as explained in the technical overview, the reduction has to rely on a small γ_{min}. As it is clear from Theorem 1, a smaller γ_{min} results in a worser reduction.

It is worth recalling that we cannot choose an arbitrary approximation $\widetilde{\gamma}(\vec{\mathsf{ID}})$, say $\widetilde{\gamma}(\vec{\mathsf{ID}})=\gamma(\vec{\mathsf{ID}})$, as $\widetilde{\gamma}(\vec{\mathsf{ID}})$ needs to be *efficiently* computable. This is because the reduction must compute $1-\gamma_{\mathsf{min}}/\widetilde{\gamma}(\vec{\mathsf{ID}})$ to perform the artificial abort.

In the remainder of this section, we propose four partitioning functions allowing to *efficiently* approximate $\gamma(\vec{\mathsf{ID}})$ better than γ_{min}. Each partitioning function has different characteristics and can be embedded into a wide class of cryptographic primitives with different algebraic properties. An overview of the partitioning functions with approximation can be found in the following Table 1. One of the four partitioning functions $\mathsf{F}_{\mathsf{ParWat}}$ is new to this work. $\mathsf{F}_{\mathsf{SSM}}$, $\mathsf{F}_{\mathsf{Wat}}$, and $\mathsf{F}_{\mathsf{Boy}}$ appear in [Lys02], [Wat05], and [ABB10a], respectively. The novelty of our work is proving that each of $\mathsf{F}_{\mathsf{SSM}}$, $\mathsf{F}_{\mathsf{Wat}}$, and $\mathsf{F}_{\mathsf{Boy}}$ has a corresponding efficiently computable approximation $\widetilde{\gamma}(\vec{\mathsf{ID}})$ better than γ_{min}, where in the case of $\mathsf{F}_{\mathsf{SSM}}$, we have to use specific error correcting codes in order for our analysis to work. A concrete example of how to use our partitioning function with approximation along with Theorem 1 is given in Sec. 6 and 7.

5.2 Definition of Partitioning Function with Approximation

We first define a partitioning function *with* approximation. The definition is based on [Yam17], where we extend the original definition to capture a finer grained approximation of γ. We recover the original definition by setting $\widetilde{\gamma}(\vec{\mathsf{x}})=\gamma_{\mathsf{min}}$.

Definition 9 (Partitioning Function with Approximation)**.** *Let* $\mathsf{F}=\left\{\mathsf{F}_\lambda:\mathcal{K}_\lambda\times\{0,1\}^{\ell(\lambda)}\to\{0,1\}\right\}_{\lambda\in\mathbb{N}}$ *be an ensemble of function families. We say that* F *is a* $(\gamma_{\mathsf{min}},T_{\mathsf{F}},T_{\mathsf{approx}})$-partitioning function*, if there exists an efficient algorithm* $\mathsf{PrtSmp}(1^\lambda,Q,\epsilon)$*, which takes as input a polynomially bounded* $Q=Q(\lambda)\in\mathbb{N}$ *and a noticeable* $\epsilon=\epsilon(\lambda)\in(0,1/2]$ *and outputs a* partitioning key K *such that:*

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

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

Table 1: Different Types of Partitioning Function and their Quality of γ_{min}.

Partitioning Function	γ_{min} with [BR09] Analysis	γ_{min} with Fine-tuned Analysis	Misc.
$\mathsf{F}_{\mathsf{Wat}}$ (Sec. 5.3)	$O(\epsilon/\ell Q)$	$O(\sqrt{\epsilon}/\ell Q)$	pairing: IBEs and VRFs lattice: IBE with exp. modulus q
$\mathsf{F}_{\mathsf{Boy}}$ (Sec. 5.4)	$O(\epsilon^2/Q^2)$	$O(\epsilon/Q^2)$	lattice IBEs
$\mathsf{F}_{\mathsf{ParWat}}$ (Sec. 5.5)	$O(\epsilon/qQ)$	$O(\epsilon^{1/d}/qQ)^{\dagger}$	lattice IBEs
$\mathsf{F}_{\mathsf{SSM}}$ (Sec. 5.6, Binary)	$O((\epsilon/Q)^{\mu})$	$O((\sqrt{\epsilon}/Q)^{\mu})$	pairing and lattice IBEs & VRFs
$\mathsf{F}_{\mathsf{SSM}}$ (Sec. 5.6, Poly)	$O((\epsilon/\ell Q)^{1+1/\nu})^{\ddagger}$	$O(\sqrt{\epsilon}/\ell^{\nu}Q)$	pairing and lattice IBEs & VRFs

The table shows four different partitioning functions. A black (resp., gray) entry shows that the corresponding bound is proven in our work (resp., previous work). The column "γ_{min} with [BR09] Analysis" shows lower bounds on γ_{min} derived from Bellare-Ristenpart-style analysis, where $\widetilde{\gamma}(\vec{\mathsf{x}})$ is a fixed value that does not depend on $\vec{\mathsf{x}}$. The column "γ_{min} with Fine-tuned Analysis" shows lower bounds on γ_{min} derived from our fine-tuned analysis, where $\widetilde{\gamma}$ can be dependent on the input $\vec{\mathsf{x}}$. For $\mathsf{F}_{\mathsf{SSM}}$, "Binary" (resp., "Poly") represents the case where the underlying error correcting code is instantiated over binary (resp., polynomial size) alphabet. In the table, ℓ is the length of the input, q is the size of the modulus used in the lattice based constructions, and d is an integer that can be set arbitrarily, which is determined by the underlying hash functions. The constants $\mu > 1$ and $1 \geq \nu > 0$ are determined by the underlying error correcting codes and can be set arbitrarily.

† By choosing $d = \omega(1)$, we can achieve $\gamma_{\mathsf{min}} = O(1/q\lambda Q)$, which removes the dependency on ϵ altogether.

‡ The bound here is due to Kohl [Koh19]. We can improve the bound to $O(\epsilon/\ell^{\nu}Q)$ using our error correcting code. We refer to Remark 5 for the details.

for all $\lambda > \lambda_0$. Here, λ_0 may depend on functions $Q(\lambda)$ and $\epsilon(\lambda)$.

2. *For a vector $\vec{\mathsf{x}} := (\mathsf{x}^*, \mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(Q)}) \in (\{0,1\}^{\ell})^{Q+1}$, let us define $\gamma(\lambda, \vec{\mathsf{x}})$ as*

$$\gamma(\lambda, \vec{\mathsf{x}}) := \Pr\left[\mathsf{F}(K, \mathsf{x}^{(1)}) = \cdots = \mathsf{F}(K, \mathsf{x}^{(Q)}) = 1 \wedge \mathsf{F}(K, \mathsf{x}^*) = 0 : K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^{\lambda}, Q(\lambda), \epsilon(\lambda)\right)\right].$$

For $\lambda > \lambda_0$, there exist $\gamma_{\mathsf{min}}(\lambda)$ and $\widetilde{\gamma}(\lambda, \vec{\mathsf{x}})$ that depend on $Q(\lambda)$ and $\epsilon(\lambda)$ such that for all distinct $\mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(Q)}, \mathsf{x}^* \in \{0,1\}^{\ell}$, the following hold:

$$\gamma(\lambda, \vec{\mathsf{x}}) \geq \gamma_{\min}(\lambda),\ \widetilde{\gamma}(\lambda, \vec{\mathsf{x}}) \geq \gamma_{\min}(\lambda),\ \ |\gamma(\lambda, \vec{\mathsf{x}}) - \widetilde{\gamma}(\lambda, \vec{\mathsf{x}})| < \frac{\gamma_{\min}(\lambda)}{3} \cdot \epsilon. \tag{25}$$

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

3. *For $\lambda > \lambda_0$, there exists an algorithm that takes λ, Q, ϵ, and $\vec{\mathsf{x}}$ as input and computes $\gamma_{\mathsf{min}}(\lambda)$ and $\widetilde{\gamma}(\lambda, \vec{\mathsf{x}})$ in time $T_{\mathsf{approx}}(\lambda, Q, \epsilon)$. Moreover, for all $\lambda > \lambda_0$, $K \in \mathcal{K}$ and $\mathsf{x} \in \{0,1\}^{\ell}$, $\mathsf{F}(K, \mathsf{x})$ can be computed in time $T_{\mathsf{F}}(\lambda)$.*

We may drop the subscript λ and denote F, $\mathcal{K}$, and $\mathcal{X}$ for the sake of simplicity.

5.3 Partitioning Function Underlying Waters IBE

Here, we analyze the partitioning function $\mathsf{F}_{\mathsf{Wat}}$ used by Waters [Wat05]. Due to its algebraic simplicity, this has been successfully used in many other constructions such as [BMW05, ABB10a, HW10, KPC+11, DKPW12]. Formally, $\mathsf{F}_{\mathsf{Wat}}$ is defined as follows:

$$\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}) = \begin{cases} 0 & K_0 + \sum_{i:\mathsf{x}_i=1} K_i = 0 \mod p \\ 1 & \text{otherwise} \end{cases}$$

where $K := (K_0, K_1, \ldots, K_\ell) \in \mathcal{K} := [-(p-1)/2, (p-1)/2]^{\ell+1}$, $\mathsf{x} \in \{0,1\}^\ell$, x_i is the i-th bit of $\mathsf{x} \in \{0,1\}^\ell$, and p is a prime integer.

The following theorem provides a more fine-grained analysis of $\mathsf{F}_{\mathsf{Wat}}$ compared to prior works.

Theorem 2. *Let $p = p(\lambda) \geq 2^\lambda$ be a prime, $\epsilon = \epsilon(\lambda)$ be a noticeable function in $(0, 1/2]$, and $\ell = \ell(\lambda)$ and $Q = Q(\lambda)$ be a polynomially bounded positive integers such that $Q \leq p\sqrt{\epsilon}/\ell\sqrt{3}$. Then, $\mathsf{F}_{\mathsf{Wat}}$ is a $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$-partitioning function with approximation such that*

$$\gamma_{\mathsf{min}} = \frac{1}{\ell N + 1}\left(1 - \frac{Q}{N+1}\right),\ T_{\mathsf{F}} = \ell \cdot \mathsf{poly}(\lambda),\ \text{and}\ T_{\mathsf{approx}} = (Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda)$$

where $N = \lfloor \sqrt{3} \cdot Q/\sqrt{\epsilon} \rfloor$ and $\mathsf{poly}(\lambda)$ is a fixed polynomial independent from Q and ϵ. In particular, we have $\gamma_{\mathsf{min}} > \sqrt{\epsilon_{\mathsf{A}}}/7Q\ell$.

Proof. We first define the algorithm $\mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$.

$\mathsf{PrtSmp}(1^\lambda, Q, \epsilon) \to K$**:** It takes as input a security parameter 1^λ, a polynomial bounded $Q = Q(\lambda)$, and a noticeable $\epsilon = \epsilon(\lambda) \in (0, 1/2]$. It defines $N := \lfloor \sqrt{3} \cdot Q/\sqrt{\epsilon} \rfloor$, samples $K \xleftarrow{\$} [-\ell N, 0] \times [0, N]^\ell$, and returns K.

It is clear that PrtSmp terminates in polynomial time. Below, we show that PrtSmp satisfies the three properties in Def. 9.

First property. We start with the first property. When Q and ℓ are polynomially bounded and ϵ is noticeable, we have

$$\ell N \leq \ell Q \sqrt{\frac{3}{\epsilon}} = \mathsf{poly}(\lambda).$$

Since $p \geq 2^\lambda$, we have $[-\ell N, 0] \times [0, N]^\ell \subset [-(p-1)/2, (p-1)/2]^{\ell+1} = \mathcal{K}$ for sufficiently large λ. Since the output K of PrtSmp is always included in $\mathcal{K}$, PrtSmp satisfies the first property.

Second property. Below, for simplicity, we omit λ when the context is clear. For $\vec{\mathsf{x}} = (\mathsf{x}^*, \mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(Q)})$, define $\gamma(\vec{\mathsf{x}})$ as

$$\gamma(\vec{\mathsf{x}}) := \Pr\left[\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^{(1)}) = \cdots = \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^{(Q)}) = 1 \wedge \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0\right],$$

where the probability is taken over the choice of $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right)$.

Further define γ_{min} and $\widetilde{\gamma}(\vec{\mathsf{x}})$ as

$$\gamma_{\mathsf{min}} := \frac{1}{\ell N + 1}\left(1 - \frac{Q}{N+1}\right) \text{ and} \tag{26}$$

$$\widetilde{\gamma}(\vec{\mathsf{x}}) := \Pr[\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0] - \sum_{j \in [Q]} \Pr[\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^{(j)}) = 0]. \tag{27}$$

Below, we show that $\gamma(\vec{\mathsf{x}})$, γ_{min}, and $\widetilde{\gamma}(\vec{\mathsf{x}})$ satisfy the three inequalities in Def. 9, Item 2.

Let us first focus on the first inequality: $\gamma(\vec{\mathsf{x}}) \geq \gamma_{\mathsf{min}}$. Notice that if $\gamma(\vec{\mathsf{x}}) \geq \widetilde{\gamma}(\vec{\mathsf{x}})$, then the second inequality in Def. 9, Item 2 implies the first inequality. Since we will show the second

inequality later, we only need to show $\gamma(\vec{\mathsf{x}}) \geq \widetilde{\gamma}(\vec{\mathsf{x}})$. Let $\mathsf{E}(\mathsf{x})$ be the event that $\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}) = 0$ holds where $K \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$. Then, we have

$$\begin{aligned}
\gamma(\vec{\mathsf{x}}) &= \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \neg\mathsf{E}(\mathsf{x}^{(1)}) \wedge \cdots \wedge \neg\mathsf{E}(\mathsf{x}^{(Q)})] \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \neg(\neg\mathsf{E}(\mathsf{x}^{(1)}) \wedge \cdots \wedge \neg\mathsf{E}(\mathsf{x}^{(Q)}))] \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr[\mathsf{E}(\mathsf{x}^*) \wedge (\mathsf{E}(\mathsf{x}^{(1)}) \vee \cdots \vee \mathsf{E}(\mathsf{x}^{(Q)}))] \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr[(\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(1)})) \vee \cdots \vee (\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(Q)}))] \\
&\geq \Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j\in[Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] = \widetilde{\gamma}(\vec{\mathsf{x}}),
\end{aligned}$$

where the third equation follows from the De Morgan's laws and the final inequality follows from the union bound. Thus, $\gamma(\vec{\mathsf{x}}) \geq \widetilde{\gamma}(\vec{\mathsf{x}})$ as desired.

We next show the second inequality: $\widetilde{\gamma}(\vec{\mathsf{x}}) \geq \gamma_{\mathsf{min}}$. From $Q \leq p\sqrt{\epsilon}/\ell\sqrt{3}$ and $N \leq \sqrt{3} \cdot Q/\sqrt{\epsilon}$, we have $\ell N \leq p$. Because there is exactly one $K_0 \in [\ell N, 0]$ satisfying $\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0$ for any $\{\mathsf{x}_i\}_{i\in[\ell]} \in [0, N]^\ell$ and K_0 is chosen uniformly at random from $[-\ell N, 0]$, we have $\Pr[\mathsf{E}(\mathsf{x}^*)] = 1/(\ell N + 1)$. From this fact, we have

$$\widetilde{\gamma}(\vec{\mathsf{x}}) = \frac{1}{\ell N + 1} - \sum_{j\in[Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] \tag{28}$$

Now, we derive an upper bound of $\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$ for any $j \in [Q]$. Let $S(\mathsf{x}) \subseteq [\ell]$ be a set of indices such that $\mathsf{x}_i = 1$. First, we consider the case $|S(\mathsf{x}^*)| \leq |S(\mathsf{x}^{(j)})|$. Because $\mathsf{x}^* \neq \mathsf{x}^{(j)}$, there is at least one index $k \in [\ell]$ such that $k \in S(\mathsf{x}^{(j)})$ and $k \notin S(\mathsf{x}^*)$. Then, we have

$$\begin{aligned}
&\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] \\
&= \Pr[\mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0] \cdot \Pr\left[\sum_{i\in S(\mathsf{x}^{(j)})} K_i = \sum_{i\in S(\mathsf{x}^*)} K_i \middle| \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0\right] \\
&= \frac{1}{\ell N + 1} \cdot \Pr\left[\sum_{i\in S(\mathsf{x}^{(j)})} K_i = \sum_{i\in S(\mathsf{x}^*)} K_i \middle| \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0\right] \\
&= \frac{1}{\ell N + 1} \cdot \Pr\left[K_k = \sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i \middle| \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0\right] \\
&= \frac{1}{\ell N + 1} \cdot \sum_{a\in[0,N]} \Pr\left[K_k = a \wedge \sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i = a \middle| \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0\right] \\
&= \frac{1}{\ell N + 1} \cdot \sum_{a\in[0,N]} \Pr\left[\sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i = a \middle| \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0\right] \\
&\qquad \times \Pr\left[K_k = a \middle| \mathsf{F}_{\mathsf{Wat}}(K, \mathsf{x}^*) = 0 \wedge \sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i = a\right].
\end{aligned} \tag{29}$$

Because $k \notin S(\mathsf{x}^*)$, the event $(\mathsf{F}_{\mathsf{Wat}}(K,\mathsf{x}^*)=0) \wedge (\sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i = a)$ is independent of the event $K_k = a$. Since K_k is chosen uniformly at random from $[0,N]$, $\Pr[K_k = a | \mathsf{F}_{\mathsf{Wat}}(K,\mathsf{x}^*)=0 \wedge \sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i = a]$ is equal to $1/(N+1)$. Therefore, we obtain

$$\begin{aligned}
\text{Eq. (29)} &= \frac{1}{\ell N+1} \cdot \sum_{a\in[0,N]} \Pr\left[\sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i = a \middle| \mathsf{F}_{\mathsf{Wat}}(K,\mathsf{x}^*)=0\right] \cdot \frac{1}{N+1} \\
&= \frac{1}{(\ell N+1)(N+1)} \cdot \Pr\left[\sum_{i\in S(\mathsf{x}^*)} K_i - \sum_{i\in S(\mathsf{x}^{(j)})\setminus\{k\}} K_i \in [0,N] \middle| \mathsf{F}_{\mathsf{Wat}}(K,\mathsf{x}^*)=0\right] \\
&\le \frac{1}{(\ell N+1)(N+1)}. \qquad (30)
\end{aligned}$$

The other case $|S(\mathsf{x}^*)| > |S(\mathsf{x}^{(j)})|$ follows similarly and we obtain the same inequality. Applying this to Eq. (28), we have

$$\tilde{\gamma}(\vec{\mathsf{x}}) \ge \frac{1}{\ell N+1} - \sum_{j\in[Q]} \frac{1}{(\ell N+1)(N+1)} = \frac{1}{\ell N+1}\left(1 - \frac{Q}{N+1}\right) = \gamma_{\mathsf{min}}.$$

This establishes the second inequality.

Finally, we show the third inequality: $|\gamma(\lambda,\vec{\mathsf{x}}) - \widetilde{\gamma}(\lambda,\vec{\mathsf{x}})| < \frac{\gamma_{\min}(\lambda)}{3}\cdot\epsilon$. Recall we have

$$\gamma(\vec{\mathsf{x}}) = \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \neg\mathsf{E}(\mathsf{x}^{(1)}) \wedge \cdots \wedge \neg\mathsf{E}(\mathsf{x}^{(Q)})] = \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr\left[\bigvee_{j\in[Q]} (\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}))\right].$$

By Bonferroni's inequalities, we obtain the following bound.

$$\begin{aligned}
&\Pr\left[\bigvee_{j\in[Q]} (\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}))\right] \\
&\ge \sum_{j\in[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})\right] - \sum_{1\le j<k\le Q} \Pr\left[(\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})) \wedge (\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(k)}))\right] \\
&= \sum_{j\in[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})\right] - \sum_{1\le j<k\le Q} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right].
\end{aligned}$$

Thus, we have

$$\gamma(\vec{\mathsf{x}}) \le \Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j\in[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})\right] + \sum_{1\le j<k\le Q} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right].$$

From Eq. (27) and $\tilde{\gamma}(\vec{\mathsf{x}}) \le \gamma(\vec{\mathsf{x}})$, we have

$$|\gamma(\vec{\mathsf{x}}) - \tilde{\gamma}(\vec{\mathsf{x}})| \le \sum_{1\le j<k\le Q} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right]. \qquad (31)$$

We use the following two lemmas to derive an upper bound for $\Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right]$ for any $1 \le j < k \le Q$. So as not to interrupt the proof, the proofs are postponed to the end.

Lemma 2. *Let ℓ and p be positive integers such that $\ell \geq 3$ and $p \geq 3$ a prime. Further, let $\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3 \in \{0,1\}^\ell$ be arbitrary but mutually distinct vectors. Then, the following matrix $\mathbf{A}$ is full rank over modulo p.*

$$\mathbf{A} := \begin{bmatrix} 1 & \mathsf{x}_1 \\ 1 & \mathsf{x}_2 \\ 1 & \mathsf{x}_3 \end{bmatrix} \subset \mathbb{Z}_p^{3\times(\ell+1)}.$$

Lemma 3. *Let ℓ, N, and p be positive integers such that $\ell \geq 3$ and $p \geq 3$ a prime. Further, let $\mathbf{A} \in \mathbb{Z}_p^{3\times(\ell+1)}$ be an arbitrary full-rank matrix such that $\mathbf{A}_{1,1} \neq 0 \mod p$ (i.e., the top left entry is non-zero). Then, we have the following.*

1. *If $\ell N < p$ and we sample a row vector $K \xleftarrow{\$} [0, \ell N + 1] \times [0, N]^\ell$, then $\Pr[\mathbf{A}K^\top = 0 \mod p] \leq \frac{1}{(\ell N+1)(N+1)^2}$.*
2. *If we sample a row vector $K \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$, $\Pr[\mathbf{A}K^\top = 0 \mod p] = \frac{1}{p^3}$.*

Using these two lemmas, the upper bound follows naturally. Let us set $(\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3)$ in Lemma 2 as $(\mathsf{x}^*, \mathsf{x}^{(j)}, \mathsf{x}^{(k)})$ for any $1 \leq j < k \leq Q$ and invoke Lemma 3, Item 1. We then obtain $\Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right] \leq 1/(\ell N + 1)(N + 1)^2$ and arrive at the following:

$$\begin{aligned} |\gamma(\vec{\mathsf{x}}) - \tilde{\gamma}(\vec{\mathsf{x}})| &\leq \sum_{1\leq j<k\leq Q} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right] \\ &\leq \frac{Q^2}{2(\ell N + 1)(N + 1)^2}. \end{aligned}$$

It remains to show the following inequality for the third inequality.

$$\frac{Q^2}{2(\ell N + 1)(N + 1)^2} < \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon.$$

Plugging in $\gamma_{\mathsf{min}} = (1 - Q/(N + 1))/(\ell N + 1)$, this is equivalent to showing the following:

$$\frac{Q^2}{2(N + 1)^2} < \left(1 - \frac{Q}{N + 1}\right) \cdot \frac{\epsilon}{3}.$$

Since $\epsilon \in (0, 1/2]$, $N \leq \sqrt{3} \cdot Q/\sqrt{\epsilon}$ implies $Q \geq N/\sqrt{6}$. Moreover, since $N = \lfloor \sqrt{3} \cdot Q/\sqrt{\epsilon} \rfloor$, we have $\epsilon > 3Q^2/(N + 1)^2$. By substituting $\epsilon > 3Q^2/(N + 1)^2$ to the right hand side, we have

$$\text{(r.h.s.)} = \left(1 - \frac{Q}{N + 1}\right) \cdot \frac{Q^2}{(N + 1)^2} > \left(1 - \frac{N}{\sqrt{6} \cdot (N + 1)}\right) \cdot \frac{Q^2}{(N + 1)^2} > \frac{Q^2}{2(N + 1)^2} \tag{32}$$

where the first inequality follows from $Q \leq N/\sqrt{6}$, the second inequality follows from the fact that $2(1 - N/\sqrt{6} \cdot (N + 1)) > 1/2$. This establishes the third inequality.

Combining everything, $\mathsf{F}_{\mathsf{Wat}}$ indeed satisfies the second property of Def. 9. Plugging $\sqrt{3} \cdot$

$Q/\sqrt{\epsilon_{\mathsf{A}}} - 1 < N \leq \sqrt{3} \cdot Q/\sqrt{\epsilon_{\mathsf{A}}}$ into $\gamma_{\mathsf{min}} = (1 - Q/(N+1))/(\ell N + 1)$, we have

$$\begin{aligned}\frac{1}{(\ell N+1)}\left(1-\frac{Q}{N+1}\right) &> \frac{1}{(\ell(\sqrt{3}\cdot Q/\sqrt{\epsilon_{\mathsf{A}}})+1)}\left(1-\frac{Q}{\sqrt{3}\cdot Q/\sqrt{\epsilon_{\mathsf{A}}}}\right)\\ &> \frac{\sqrt{\epsilon_{\mathsf{A}}}}{(\ell\sqrt{3}\cdot Q+\sqrt{\epsilon_{\mathsf{A}}})}\left(1-\frac{1}{\sqrt{3}}\right)\\ &> \frac{\sqrt{\epsilon_{\mathsf{A}}}}{(6\ell Q+2\sqrt{3\epsilon_{\mathsf{A}}})} > \frac{\sqrt{\epsilon_{\mathsf{A}}}}{7Q\ell}\end{aligned}$$

where the third inequality follows $1 - 1/\sqrt{3} > 1/2\sqrt{3}$. Thus we have $\gamma_{\mathsf{min}} > \sqrt{\epsilon_{\mathsf{A}}}/7Q\ell$ as in the theorem statement.

Third property. Finally, we show the third property of Def. 9. It is clear that γ_{min} can be computable in time $\mathsf{poly}(\log Q, \log(1/\epsilon))$ which is upper bounded by a fixed polynomial since Q is a polynomial and ϵ is noticeable. While we can naively compute $\widetilde{\gamma}(\vec{\mathsf{x}})$ using time $Q \cdot \mathsf{poly}(N) = \mathsf{poly}(Q, 1/\epsilon)$, we want to avoid this. This is because when we consider applications of our analysis to IBEs and VRFs, having large T_{approx} leads to large runtime loss in the reductions and ruins the advantage of having larger advantages, when we consider the overall reduction cost. Luckily, we can do much better. Namely, we show in Sec. 8, Theorem 15 that there exists an algorithm that takes as input λ, Q, ϵ, and $\vec{\mathsf{x}}$ and computes $\widetilde{\gamma}(\vec{\mathsf{x}})$ in time only $(Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda)$, i.e., independent of ϵ and linear in Q. Thus, $T_{\mathsf{approx}} = (Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda)$. Lastly, since it requires ℓ addition and one modulo $p \approx 2^\lambda$ operation to compute $\mathsf{F}_{\mathsf{Wat}}$, we have $T_{\mathsf{F}} = \ell \cdot \mathsf{poly}(\lambda)$. This completes the proof of the third property.

Lastly, we prove the postponed proof of Lemmas 2 and 3 below.

Proof of Lemma 2. Since $p \geq 3$ and $(\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3)$ are mutually distinct, $\mathrm{rank}\,(\mathbf{A}) \neq 1$ cannot occur. For the sake of contradiction, suppose $\mathrm{rank}\,(\mathbf{A}) = 2$. Then, there exists a pair $(a, b) \in \mathbb{Z}_p^2$ such that $(1||\mathsf{x}_1) = a(1||\mathsf{x}_2) + b(1||\mathsf{x}_3) \mod p$. If $a = 0 \mod p$, then $(1||\mathsf{x}_1) = b(1||\mathsf{x}_2) \mod p$. While this implies $b = 1 \mod p$, it contradicts $\mathsf{x}_1 \neq \mathsf{x}_2$. Thus, we can assume $a \neq 0 \mod p$. Similarly, we can assume $b \neq 0 \mod p$. Next, looking at the first entry of the equality, we have $a + b = 1 \mod p$. Combined with $a, b \neq 0 \mod p$, we have $a, b \neq 1 \mod p$, that is, $a, b \in \mathbb{Z}_p \backslash \{0, 1\}$. Now, since $\mathsf{x}_2, \mathsf{x}_3 \in \{0,1\}^\ell$ are distinct, $a\mathsf{x}_2 + b\mathsf{x}_3$ must include an entry that is either a or b. However, since $a(1||\mathsf{x}_2) + b(1||\mathsf{x}_3) \mod p = (1||\mathsf{x}_1) \in \{0,1\}^\ell$, this implies either $a = 1$ or $b = 1$, thus contradicting $a, b \in \mathbb{Z}_p \backslash \{0, 1\}$. Therefore, we conclude $\mathrm{rank}\,(\mathbf{A}) \neq 2$. Thus, we arrive at $\mathrm{rank}\,(\mathbf{A}) = 3$. □

Proof of Lemma 3. By Gaussian elimination, there exist a matrix $\mathbf{L} \in \mathbb{Z}_p^{3\times 3}$, a permutation matrix $\mathbf{P} \in \mathbb{Z}_p^{(\ell+1)\times(\ell+1)}$, and a matrix $\mathbf{B} \in \mathbb{Z}_p^{3\times(\ell-2)}$ such that $\mathbf{A} = \mathbf{L}[\mathbf{I}_3|\mathbf{B}]\mathbf{P} \mod p$ where $\mathbf{I}_3$ is the identity matrix of size 3. Notice that $\mathbf{L}$ is non-singular because $\mathbf{A}$ is full rank. Since $\mathbf{A}_{1,1}$ is non-zero, we can assume that the first column (resp. row) of $\mathbf{P}$ is $(1, 0, \ldots, 0)^\top$ (resp. $(1, 0, \ldots, 0)$).

We first focus on Item 1. Let us set $K' := K\mathbf{P}^\top$. Then, since $\mathbf{P}$ is a permutation that keeps the first entry in place, K' is distributed identically to $K \xleftarrow{\$} [-\ell N, 0] \times [0, N]^\ell$. Let $K'_{\leq 2}$ be the first three elements in K' and $K'_{>2}$ be elements in K' after the third one. (Namely, for

$K' = (K_0, K'_1, \ldots, K'_\ell)$, $\bar{K'}_{\leq 2} = (K_0, K'_1, K'_2)$ and $K'_{>2} = (K'_3, \ldots, K'_\ell)$.) Then, we have

$$\begin{aligned}
& \mathbf{A}K^\top = 0 \mod p \\
\Rightarrow \quad & L[\mathbf{I}_3|\mathbf{B}]\mathbf{P}K^\top = 0 \mod p \\
\Rightarrow \quad & L[\mathbf{I}_3|\mathbf{B}]K'^\top = 0 \mod p \\
\Rightarrow \quad & [\mathbf{I}_3|\mathbf{B}]K'^\top = 0 \mod p \\
\Rightarrow \quad & \mathbf{I}_3 K'^\top_{\leq 2} + \mathbf{B}K'^\top_{>2} = 0 \mod p \\
\Rightarrow \quad & K'^\top_{\leq 2} = -\mathbf{B}K'^\top_{>2} \mod p.
\end{aligned}$$

The third change is true because L is non-singular. Since $K'_{\leq 2}$ is uniformly distributed over $[-\ell N, 0] \times [0, N]^2$ independently of $K'^\top_{>2}$ and $\ell N \leq p$, $\mathbf{A}K^\top = 0 \mod p$ holds with probability at most $1/(\ell N + 1)(N+1)^2$ as desired.

The case where $K \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$ follows an identical argument. The only difference is that the we have exactly $\frac{1}{p^3}$ rather than an upper bound. This follows from the fact that $K'_{\leq 2}$ is uniformly distributed over $\mathbb{Z}_p^3$. This concludes the proof. □

□

5.4 Partitioning Function Underlying ABB IBE

We next analyze the partitioning function $\mathsf{F}_{\mathsf{Boy}}$ originally used by Boyen [Boy10] to construct lattice-based signatures, and then subsequently used by Agrawal, Boneh, and Boyen [ABB10a] to construct lattice-based IBE schemes. While Waters' partitioning function $\mathsf{F}_{\mathsf{Wat}}$ can, in principle, be used to construct lattice-based signatures and IBE schemes, it requires the modulus q to be exponential, leading to a large inefficiency. To this end, Boyen devised a partitioning function more suited to the algebraic constraints of lattice.

Let n, k, q be integers such that $k|n$ and q a prime. Let $\mathsf{H}^{\mathsf{frd}}$ be a full-rank difference encoding as defined in Def. 18, which takes a vector in $\mathbb{Z}_q^j$ with arbitrary j and outputs a matrix in size $\mathbb{Z}_q^{j \times j}$. Formally, $\mathsf{F}_{\mathsf{Boy}}$ is defined as follows:

$$\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}) = \begin{cases} 0 & \left(\mathsf{H}^{\mathsf{frd}}(K_0) + \sum_{i:\mathsf{x}_i=1} \mathsf{H}^{\mathsf{frd}}(K_i)\right) \otimes \mathbf{I}_{n/k} = \mathbf{0}_{n \times n} \mod q \\ 1 & \text{otherwise} \end{cases}$$

where $K := (K_0, K_1, \ldots, K_\ell) \in \mathcal{K} := \cup_{j|n} \left(\mathbb{Z}_q^j\right)^{\ell+1}$, $\mathsf{x} \in \{0,1\}^\ell$, and x_i is the i-th bit of an identity $\mathsf{x} \in \{0,1\}^\ell$.[10]

The following theorem provides a more fine-grained analysis of $\mathsf{F}_{\mathsf{Boy}}$ compared to prior works.

Theorem 3. *Let $n = n(\lambda), \ell = \ell(\lambda), q = q(\lambda)$ be integers such that q is a prime satisfying $q/2 > \ell$. Let $\epsilon = \epsilon(\lambda)$ be a noticeable function in $(0, 1/2]$, $Q = Q(\lambda)$ be a polynomially bounded positive integer, and k be the smallest integer such that $k|n$ and $q^k \geq 2 \cdot Q \cdot \sqrt{\epsilon}^{-1}$. Then, $\mathsf{F}_{\mathsf{Boy}}$ is a $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$-partitioning function such that*

$$\gamma_{\mathsf{min}} = \frac{1}{q^k}\left(1 - \frac{Q}{q^k}\right), \ T_{\mathsf{F}} = \ell \cdot \mathsf{poly}(\lambda), \ \textit{and} \ T_{\mathsf{approx}} = \mathsf{poly}(\lambda),$$

[10] Here, we note that the above $\mathsf{F}_{\mathsf{Boy}}$ is slightly different from the one originally defined by Boyen [Boy10]. In his work, $\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}) = 0$ if and only if $\mathbf{I}_n + \sum_{i \in [\ell]} (-1)^{\mathsf{x}_i} \cdot \mathsf{H}^{\mathsf{frd}}(K_i) \otimes \mathbf{I}_{n/k} = \mathbf{0}_{n \times n} \mod q$. It turns out that the our definition is more natural and allows for a simpler and tighter analysis.

where $\mathsf{poly}(\lambda)$ *is a fixed polynomial independent from* Q *and* ϵ*. In particular, this implies* $\gamma_{\mathsf{min}} \geq \frac{\epsilon}{8 \cdot Q^2}$.

Proof. We first define the algorithm $\mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$.

$\mathsf{PrtSmp}(1^\lambda, Q, \epsilon) \to K$**:** It takes as input a security parameter 1^λ, a polynomial bounded $Q = Q(\lambda)$, and a noticeable $\epsilon = \epsilon(\lambda) \in (0, 1/2]$. It computes the smallest integer k that satisfies $k|n$ and $q^k \geq 2 \cdot Q \cdot \sqrt{\epsilon}^{-1}$, samples $K \xleftarrow{\$} \left(\mathbb{Z}_q^k\right)^{\ell+1}$, and returns K.

It is clear that PrtSmp terminates in polynomial time. Below, we show that PrtSmp satisfies the three properties in Def. 9.

First property. It is clear that $K \in \mathcal{K} := \cup_{j|n} \left(\mathbb{Z}_q^j\right)^{\ell+1}$. Since the output K of PrtSmp is always included in $\mathcal{K}$, PrtSmp satisfies the first property.

Second property. Below, for simplicity, we omit λ when the context is clear. For $\vec{\mathsf{x}} = (\mathsf{x}^*, \mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(Q)})$, we define $\gamma(\vec{\mathsf{x}})$ as

$$\gamma(\lambda, \vec{\mathsf{x}}) := \Pr\left[\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}^{(1)}) = \cdots = \mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}^{(Q)}) = 1 \wedge \mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}^*) = 0\right]$$

where the probability is taken over the choice of $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right)$.

Further define γ_{min} and $\widetilde{\gamma}(\vec{\mathsf{x}})$ as

$$\begin{aligned}
\gamma_{\mathsf{min}} &:= \frac{1}{q^k}\left(1 - \frac{Q}{q^k}\right) \text{ and} \\
\widetilde{\gamma}(\vec{\mathsf{x}}) &:= \Pr[\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}^*) = 0] - \sum_{j \in [Q]} \Pr[\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}^*) = \mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}^{(j)}) = 0]. \qquad (33)
\end{aligned}$$

Below, we show that $\gamma(\vec{\mathsf{x}})$, γ_{min}, and $\widetilde{\gamma}(\vec{\mathsf{x}})$ satisfy the three inequalities in Def. 9, Item 2.

Using the same argument made in Theorem 2, we only need to show the second inequality as it implies the first inequality: $\widetilde{\gamma}(\vec{\mathsf{x}}) \geq \gamma_{\mathsf{min}}$. We therefore focus on the second inequality: $\widetilde{\gamma}(\vec{\mathsf{x}}) \geq \gamma_{\mathsf{min}}$. First, observe that since $\mathsf{F}_{\mathsf{Boy}}$ induces a matrix that replicates the same matrix along the diagonal n/k times, $\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}) = 0$ if and only if $f_{\mathsf{x}}(K) := \mathsf{H}_k^{\mathsf{frd}}(K_0) + \sum_{i:\mathsf{x}_i=1} \mathsf{H}_k^{\mathsf{frd}}(K_i) = \mathbf{0}_{k \times k} \mod q$. Now, since $\mathsf{H}_k^{\mathsf{frd}}$ is linearly homomorphic and $\mathbf{0}_k$ is the only vector that gets mapped to $\mathbf{0}_{k \times k}$ by $\mathsf{H}_k^{\mathsf{frd}}$, $f_{\mathsf{x}}(K) = \mathbf{0}_{k \times k}$ if and only if $K_0 + \sum_{i:\mathsf{x}_i} K_i = \mathbf{0}_k$. Furthermore, since each entry of $K_0, K_1, \ldots, K_\ell$ is distributed independently of each other, we can analyze the probability that $f_{\mathsf{x}}(K) = \mathbf{0}_{k \times k}$ entry-wise. That is, for any $\mathsf{x} \in \{0,1\}^\ell$, we have $\Pr[f_{\mathsf{x}}(K) = \mathbf{0}_{k \times k}] = \prod_{j \in [k]} \Pr\left[K_0[j] + \sum_{i:\mathsf{x}_i} K_i[j] = 0\right]$, where $K_i[j]$ denotes the j-th entry of $K_i \in \mathbb{Z}_q^k$. Let $\mathsf{E}(\mathsf{x})$ be the event that $\mathsf{F}_{\mathsf{Boy}}(K, \mathsf{x}) = 0$ for $K \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$. Due to the above argument, we only need to individually focus on the event $\mathsf{E}_j(\mathsf{x})$ defined as $K_0[j] + \sum_{i:\mathsf{x}_i} K_i[j] = 0$ for an arbitrary $j \in [k]$.

From Lemma 2 and Lemma 3, Item 2, it is easy to check that for any distinct $\mathsf{x}, \mathsf{x}', \mathsf{x}''$, we have the following for for any $j \in [k]$:

$$\Pr[\mathsf{E}_j(\mathsf{x})] = \frac{1}{q}, \quad \Pr[\mathsf{E}_j(\mathsf{x}) \wedge \mathsf{E}_j(\mathsf{x}')] = \frac{1}{q^2}, \quad \Pr[\mathsf{E}_j(\mathsf{x}) \wedge \mathsf{E}_j(\mathsf{x}') \wedge \mathsf{E}_j(\mathsf{x}'')] = \frac{1}{q^3}.$$

Here, the equality holds exactly as for any $(i, j) \in [\ell] \times [k]$, $K_i[j]$ is distributed uniformly at random over $\mathbb{Z}_q$.

With these preparations, we can now check the second inequality. Plugging in the value of $\Pr[\mathsf{E}(\mathsf{x}^*)]$ and $\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x})]$ into Eq. (33), we have the following for any x:

$$\widetilde{\gamma}(\mathsf{x}) = \frac{1}{q^k} - \frac{Q}{q^{2k}} = \gamma_{\mathsf{min}}.$$

This establishes the second inequality.

Finally, we show the third inequality: $|\gamma(\lambda, \vec{\mathsf{x}}) - \widetilde{\gamma}(\lambda, \vec{\mathsf{x}})| < \frac{\gamma_{\mathsf{min}}(\lambda)}{3} \cdot \epsilon$. Following an exact argument made in the proof of Theorem 2, we have

$$|\gamma(\vec{\mathsf{x}}) - \tilde{\gamma}(\vec{\mathsf{x}})| \leq \sum_{1 \leq j < k \leq Q} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)}) \wedge \mathsf{E}(\mathsf{x}^{(k)})\right].$$

As we established above, the right hand side is upper bounded by $(Q^2/2) \cdot q^{-3k}$. Thus, it suffices to establish

$$\frac{Q^2}{2q^{3k}} < \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon = \frac{\epsilon}{3q^k}\left(1 - \frac{Q}{q^k}\right) \tag{34}$$

When $Q < \frac{\sqrt{\epsilon} \cdot q^k}{2}$, the left hand side is upper bounded by $\frac{\epsilon}{8q^k}$. On the other hand, the right hand side is lower bounded by

$$\frac{\epsilon}{3q^k}\left(1 - \frac{Q}{q^k}\right) > \frac{\epsilon}{3q^k}\left(1 - \frac{\sqrt{\epsilon}}{2}\right) \geq \frac{\epsilon}{5q^k},$$

where the inequality follows from $\epsilon < 1/2$. This establishes the third inequality.

Combining everything, $\mathsf{F}_{\mathsf{Boy}}$ indeed satisfies the second property of Def. 9. As a concrete example, recall k is the smallest integer such that $k|n$ and $q^k \geq 2 \cdot Q \cdot \sqrt{\epsilon}^{-1}$. Thus, we have $2 \cdot Q \cdot \sqrt{\epsilon}^{-1} \geq q^{k/2}$, implying $4 \cdot Q^2 \cdot \epsilon^{-1} \geq q^k$ — it does not seem likely that we can show a better upper bound on q^k due to the restriction on $k|n$. Plugging the bounds in $\gamma_{\mathsf{min}} = \frac{1}{q^k}\left(1 - \frac{Q}{q^k}\right)$, we have $\gamma_{\mathsf{min}} \geq \frac{\epsilon}{8Q^2}$ as in the theorem statement.

Third property. Finally, we show the third property of Def. 9. Notice that for any x, we established $\widetilde{\gamma}(\mathsf{x}) = \gamma_{\mathsf{min}}$. Since γ_{min} can be computed in time $\mathsf{poly}(\log Q, \log(1/\epsilon))$ so can $\widetilde{\gamma}(\mathsf{x})$. Note we can upper bound $\mathsf{poly}(\log Q, \log(1/\epsilon)) = \mathsf{poly}(\lambda)$ by a fixed polynomial since Q is a polynomial and ϵ is noticeable. Moreover, $\mathsf{F}_{\mathsf{Boy}}$ can be computed with $k \times \ell$ additions so we have $T_{\mathsf{F}} = \ell \cdot \mathsf{poly}(\log Q, \log(1/\epsilon))$. Similarly this is upper bounded by $\ell \cdot \mathsf{poly}(\lambda)$ for some fixed polynomial as desired. □

Remark 2. *As far as we are aware of, our work provides the first formal analysis of the (variant of the) partitioning function $\mathsf{F}_{\mathsf{Boy}}$. Due to the subtle yet profound restriction that $k|n$, we can only bound γ_{min} by $O(\epsilon/Q^2)$, rather than the desired $O(\sqrt{\epsilon}/Q)$. Indeed, this quadratic worsening of the reduction appears even if we take the Bellare-Ristenpart type reduction [BR09] or the Waters type reduction [Wat05]. This is so because the issue is irrelevant on how well we approximate $\widetilde{\gamma}(\mathsf{x})$. Even relying on these prior reductions, our proof of Theorem 3 indicates that γ_{min} can only be lower bounded by $O(\epsilon^2/Q^2)$, rather than $O(\epsilon/Q)$, as conventionally thought.*

5.5 A New Partitioning Function for Lattices

Here, we present a new partitioning function that can be used in place of $\mathsf{F}_{\mathsf{Boy}}$. As shown in the previous section, $\mathsf{F}_{\mathsf{Boy}}$ leads to sub-optimal $\gamma_{\mathsf{min}} = O(\epsilon/Q^2)$ due to the restriction on $k|n$. The new partitioning function $\mathsf{F}_{\mathsf{ParWat}}$ can be viewed as performing parallel repetition of the Waters partitioning function $\mathsf{F}_{\mathsf{Wat}}$ with a twist, using a (perfect) *d-wise linearly independent hash function*.

Let $\mathsf{H}^{\mathsf{frd}}_n : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ a full-rank difference encoding as defined in Def. 18. For any integers d and $L_d = L(d)$, let $h_{d\text{-}\mathsf{wise}} : \{0,1\}^\ell \to \{0,1\}^{L_d}$ be a d-wise linearly independent hash function over $\mathbb{Z}_q$, that is, for any distinct $(\mathsf{x}_i)_{i\in[d]} \in (\{0,1\}^\ell)^d$, $(h_{d\text{-}\mathsf{wise}}(\mathsf{x}_i))_{i\in[d]}$ is linearly independent over $\mathbb{Z}_q$. For $d = 3$, we can define $h_{d\text{-}\mathsf{wise}}(\mathsf{x}) = (1, \mathsf{x})$, since as we have shown in Lemma 2, the map $\mathsf{x} \mapsto (1, \mathsf{x})$ is 3-wise linearly independent over $\mathbb{Z}_p$ for any primer $p \geq 3$. We postpone how to construct such a d-wise linearly independent hash function for $d > 3$ to Sec. 5.5.1. We then define our partitioning function $\mathsf{F}_{\mathsf{ParWat}}$ as follows:

$$\mathsf{F}_{\mathsf{ParWat}}(K, \mathsf{x}) = \begin{cases} 0 & \sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} \mathsf{H}^{\mathsf{frd}}_n(K_i) = \mathbf{0}_{n\times n} \pmod q \\ 1 & \text{otherwise} \end{cases}$$

where $K := (K_1, \ldots, K_{L_d}) \in \mathcal{K} := (\mathbb{Z}_q^n)^{L_d}$, $\mathsf{x} \in \{0,1\}^\ell$, and $h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i$ is the i-th bit of the hashed identity $h_{d\text{-}\mathsf{wise}}(\mathsf{x}) \in \{0,1\}^{L_d}$.

For this function, we have the following theorem. Notice that unlike for $\mathsf{F}_{\mathsf{Boy}}$, k is no longer restricted to satisfy $k|n$. This allows for a finer choice of k, leading to a better lower bound for γ_{min}.

Theorem 4. *Let $n = n(\lambda), \ell = \ell(\lambda), q = q(\lambda), d = d(\lambda)$ be integers such that q is a prime and $d \geq 3$ is odd. Let $h_{d\text{-}\mathsf{wise}} : \{0,1\}^\ell \to \{0,1\}^{L_d}$ be a d-wise linearly independent hash function over $\mathbb{Z}_q$. Let $\epsilon = \epsilon(\lambda)$ be a noticeable function in $(0, 1/2]$, $Q = Q(\lambda)$ be a polynomially bounded positive integer, let k be the smallest integer such that $q^k \geq 2 \cdot Q \cdot \epsilon^{-\frac{1}{d-1}}$. Then, $\mathsf{F}_{\mathsf{ParWat}}$ is a $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$-partitioning function such that*

$$\gamma_{\mathsf{min}} = \frac{1}{q^k} + \sum_{t\in[d-2]} (-1)^t \cdot \binom{Q}{t} \cdot \frac{1}{q^{(t+1)k}}, \ T_{\mathsf{F}} = L_d \cdot \mathsf{poly}(\lambda), \ \text{and } T_{\mathsf{approx}} = \mathsf{poly}(\lambda),$$

where $\mathsf{poly}(\lambda)$ is a fixed polynomial independent from Q and ϵ. In particular, this implies $\gamma_{\mathsf{min}} \geq \frac{\epsilon^{\frac{1}{d-1}}}{4q\cdot Q}$ and we have $\gamma_{\mathsf{min}} \geq \frac{1}{4\lambda q\cdot Q}$ if we set $d = \omega(1)$.

Proof. We first define the algorithm $\mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$.

$\mathsf{PrtSmp}(1^\lambda, Q, \epsilon) \to K$**:** It takes as input a security parameter 1^λ, a polynomial bounded $Q = Q(\lambda)$, and a noticeable $\epsilon = \epsilon(\lambda) \in (0, 1/2]$. It computes the smallest integer such $q^k \geq 2 \cdot Q \cdot \epsilon^{-\frac{1}{d-1}}$ and samples $K \xleftarrow{\$} (\mathbb{Z}_q^k \times \{0\}^{n-k})^{L_d} \subseteq (\mathbb{Z}_q^n)^{L_d}$ and returns K.

It is clear that PrtSmp terminates in polynomial time. Below, we show that PrtSmp satisfies the three properties in Def. 9.

First property. It is clear that $K \in \mathcal{K} := (\mathbb{Z}_q^n)^{L_d}$. Since the output K of PrtSmp is always included in $\mathcal{K}$, PrtSmp satisfies the first property.

Second property. For any x, denote $\mathsf{E}(\mathsf{x})$ as the event $\mathsf{F}_{\mathsf{ParWat}}(K, \mathsf{x}) = 0$. Then, for $\vec{\mathsf{x}} = (\mathsf{x}^*, \mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(Q)})$, we define $\gamma(\lambda, \vec{\mathsf{x}})$ as

$$\gamma(\lambda, \vec{\mathsf{x}}) := \Pr\left[\neg\mathsf{E}(\mathsf{x}^{(1)}) \wedge \cdots \wedge \neg\mathsf{E}(\mathsf{x}^{(Q)}) \wedge \mathsf{E}(\mathsf{x}^*)\right]$$

where the probability is taken over the choice of $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right)$.

Further define γ_{min} and $\widetilde{\gamma}(\vec{\mathsf{x}})$ as

$$\gamma_{\mathsf{min}} := \frac{1}{q^k} + \sum_{t\in[d-2]} (-1)^t \cdot \binom{Q}{t} \cdot \frac{1}{q^{(t+1)k}}$$

$$\widetilde{\gamma}(\vec{\mathsf{x}}) := \Pr[\mathsf{E}(\mathsf{x}^*)] + \sum_{t\in[d-2]} (-1)^t \cdot \left(\sum_{1\le j_1<\cdots<j_t\le[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \bigwedge_{k\in[t]} \mathsf{E}(\mathsf{x}^{(j_k)})\right]\right). \tag{35}$$

Below, we show that $\gamma(\vec{\mathsf{x}})$, γ_{min}, and $\widetilde{\gamma}(\vec{\mathsf{x}})$ satisfy the three inequalities in Def. 9, Item 2. We first make a simplifying observation: notice that for any $\vec{\mathsf{x}} \in \{0,1\}^\ell$, $\mathsf{F}_{\mathsf{ParWat}}(K, \mathsf{x}) = 0$ implies $\sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} K_i = \mathbf{0}_n \in \mathbb{Z}_q^n$ since $\mathsf{H}_n^{\mathsf{frd}}$ is linearly homomorphic and $\mathbf{0}_n$ is the only vector that gets mapped to $\mathbf{0}_{n\times n}$ by $\mathsf{H}_n^{\mathsf{frd}}$. Moreover, since each entry of $K_1, \cdots, K_n$ is distributed independently of each other, we can analyze the probability that $\sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} K_i = \mathbf{0}_n$ entry-wise. That is, for any $\mathsf{x} \in \{0,1\}^\ell$, we have $\Pr[\sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} K_i = \mathbf{0}_n] = \prod_{\nu\in[n]} \Pr\left[\sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} K_i[\nu] = 0\right] = \prod_{\nu\in[k]} \Pr\left[\sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} K_i[\nu] = 0\right]$, where $K_i[\nu]$ denotes the ν-th entry of K_i and the last equality follows from $K_i \in \mathbb{Z}_q^k \times \{0\}^{n-k}$ for all $i \in [L_d]$. For any x and $\nu \in [k]$, let us denote $\mathsf{E}_\nu(\mathsf{x})$ to be the event $\sum_{i:h_{d\text{-}\mathsf{wise}}(\mathsf{x})_i=1} K_i[\nu] = 0$ for $K \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$. Then, from the above argument, $\mathsf{E}(\mathsf{x}) = \wedge_{\nu\in[k]}\mathsf{E}_\nu(\mathsf{x})$ defines the event $\mathsf{F}_{\mathsf{ParWat}}(K, \mathsf{x}) = 0$.

Now, let us first focus on the first inequality: $\gamma(\vec{\mathsf{x}}) \ge \gamma_{\mathsf{min}}$. Notice that if $\gamma(\vec{\mathsf{x}}) \ge \widetilde{\gamma}(\vec{\mathsf{x}})$, then the second inequality in Def. 9, Item 2 implies the first inequality. Since we will show the second inequality later, we only need to show $\gamma(\vec{\mathsf{x}}) \ge \widetilde{\gamma}(\vec{\mathsf{x}})$.

$$\begin{aligned}
\gamma(\vec{\mathsf{x}}) &= \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \neg\mathsf{E}(\mathsf{x}^{(1)}) \wedge \cdots \wedge \neg\mathsf{E}(\mathsf{x}^{(Q)})] \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \neg(\neg\mathsf{E}(\mathsf{x}^{(1)}) \wedge \cdots \wedge \neg\mathsf{E}(\mathsf{x}^{(Q)}))] \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr[\mathsf{E}(\mathsf{x}^*) \wedge (\mathsf{E}(\mathsf{x}^{(1)}) \vee \cdots \vee \mathsf{E}(\mathsf{x}^{(Q)}))] \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] - \Pr[(\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(1)})) \vee \cdots \vee (\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(Q)}))] \\
&\ge \Pr[\mathsf{E}(\mathsf{x}^*)] + \sum_{t\in[d-2]} (-1)^t \cdot \left(\sum_{1\le j_1<\cdots<j_t\le[Q]} \Pr\left[\bigwedge_{k\in[t]} \left(\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j_k)})\right]\right)\right) \\
&= \Pr[\mathsf{E}(\mathsf{x}^*)] + \sum_{t\in[d-2]} (-1)^t \cdot \left(\sum_{1\le j_1<\cdots<j_t\le[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \bigwedge_{k\in[t]} \mathsf{E}(\mathsf{x}^{(j_k)})\right]\right) = \widetilde{\gamma}(\vec{\mathsf{x}}),
\end{aligned} \tag{36}$$

where the third equation follows from the De Morgan's laws and the inequality follows from the Bonferroni inequality and the fact that d is odd. Here, note that if we set $d = 3$, then the last equation becomes $\Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j\in[Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$, precisely the lower bound we used in the Waters hash $\mathsf{F}_{\mathsf{Wat}}$ in Theorem 2. In the above, we also assume implicitly that $d \le Q$; if $d = Q$, then the above will be an equality rather than an inequality. Thus, $\gamma(\vec{\mathsf{x}}) \ge \widetilde{\gamma}(\vec{\mathsf{x}})$ as desired.

We next show the second inequality: $\widetilde{\gamma}(\vec{\mathsf{x}}) \ge \gamma_{\mathsf{min}}$. Using the fact that $h_{d\text{-}\mathsf{wise}}$ is a d-wise linearly independent hash over $\mathbb{Z}_q$, for any distinct $(\mathsf{x}_i)_{i\in[d]}$, $(h_{d\text{-}\mathsf{wise}}(\mathsf{x}_i))_{i\in[d]}$ is linearly independent over $\mathbb{Z}_q$. Following an almost exact proof for Lemma 3, we have the following for every $t \in [d]$ and

$\nu \in [k]$:

$$\Pr\left[\bigwedge_{i\in[t]} \mathsf{E}_\nu(\mathsf{x}_i)\right] = \frac{1}{q^t} \quad \Rightarrow \quad \Pr\left[\bigwedge_{i\in[t]} \mathsf{E}(\mathsf{x}_i)\right] = \frac{1}{q^{tk}}, \tag{37}$$

where the implication holds from $\mathsf{E}(\mathsf{x}) = \wedge_{\nu\in[k]}\mathsf{E}_\nu(\mathsf{x})$ and the independence of $\mathsf{E}_\nu(\mathsf{x})$ for distinct ν's. Plugging this into Eq. (36), we have

$$\widetilde{\gamma}(\vec{\mathsf{x}}) = \frac{1}{q^k} + \sum_{t\in[d-2]} (-1)^t \cdot \binom{Q}{t} \cdot \frac{1}{q^{(t+1)k}} = \gamma_{\mathsf{min}}.$$

This establishes the second inequality.

Finally, we show the third inequality: $|\gamma(\lambda, \vec{\mathsf{x}}) - \widetilde{\gamma}(\lambda, \vec{\mathsf{x}})| < \frac{\gamma_{\mathsf{min}}(\lambda)}{3} \cdot \epsilon$. Following a similar argument made to derive Eq. (36), we can establish

$$\gamma(\vec{\mathsf{x}}) \le \Pr[\mathsf{E}(\mathsf{x}^*)] + \sum_{t\in[d-1]} (-1)^t \cdot \left(\sum_{1\le j_1<\cdots<j_t\le[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \bigwedge_{k\in[t]} \mathsf{E}(\mathsf{x}^{(j_k)})\right]\right),$$

where the only difference is that we use the Bonferroni inequality to upper bound, rather than lower bound, $\gamma(\vec{\mathsf{x}})$. This implies

$$|\gamma(\vec{\mathsf{x}}) - \tilde{\gamma}(\vec{\mathsf{x}})| \le \sum_{1\le j_1<\cdots<j_{d-1}\le[Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \bigwedge_{k\in[d-1]} \mathsf{E}(\mathsf{x}^{(j_k)})\right] = \binom{Q}{d-1} \cdot \frac{1}{q^{dk}},$$

where the right equality holds from Eq. (37).

It remains to show the following inequality for the third inequality.

$$\binom{Q}{d-1} \cdot \frac{1}{q^{dk}} < \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon = \frac{\epsilon}{3q^k} \left(1 + \sum_{t\in[d-2]} (-1)^t \cdot \binom{Q}{t} \cdot \frac{1}{q^{tk}}\right). \tag{38}$$

From assumption, we have $Q \le c \cdot q^k \cdot \epsilon^{1/(d-1)}$ for $c = 1/2$. Plugging this into the left hand side of Eq. (38), we have

$$(\text{l.h.s}) \le \frac{Q^{d-1}}{2^{d-2} \cdot q^{dk}} \le \frac{c^{d-1} \cdot \epsilon}{2^{d-2} \cdot q^k} = \frac{\epsilon}{2^{2d-3} \cdot q^k},$$

where the first inequality follows from the fact $(d-1)! \ge 2^{d-2}$ for $d \ge 3$. On the other hand, we have

$$\frac{\epsilon}{6q^k} \le \frac{\epsilon}{3q^k} \cdot \left(1 - c \cdot \epsilon^{\frac{1}{d-1}}\right) \le (\text{r.h.s}),$$

where the first inequality follows from $c = 1/2$, $\epsilon \in (0, 1/2]$ and $\epsilon^{1/(d-1)} < 1$ for any $d \ge 3$, and the second inequality follows implicitly from the Bonferroni inequality. Thus, for any $d \ge 3$, we have Eq. (38) as desired. This establishes the third inequality.

Combining everything, $\mathsf{F}_{\mathsf{ParWat}}$ indeed satisfies the second property of Def. 9. As a concrete example, k is the smallest integer such that $q^k \ge 2 \cdot Q \cdot \epsilon^{-\frac{1}{d-1}}$. Thus, we have $2 \cdot Q \cdot \epsilon^{-\frac{1}{d-1}} \ge q^{k-1}$,

implying $2 \cdot q \cdot Q \cdot \epsilon^{-\frac{1}{d-1}} \geq q^k$ — notice that this is a much better bound than achieved by $\mathsf{F}_{\mathsf{Boy}}$ (see proof of Theorem 3). Combined with the lower bound $\gamma_{\mathsf{min}} \geq \frac{1}{2q^k}$ (implicitly) established above, we have $\gamma_{\mathsf{min}} > \frac{\epsilon^{\frac{1}{d-1}}}{4q \cdot Q}$ as in the theorem statement. The statement on the case of $d = \omega(1)$ is obtained by observing $\epsilon > \lambda^{-d}$ for sufficiently large λ.

Third property. Finally, we show the third property of Def. 9. Notice that for any x, we established $\widetilde{\gamma}(\mathsf{x}) = \gamma_{\mathsf{min}}$. Since γ_{min} can be computed in time $\mathsf{poly}(d, \log Q, \log(1/\epsilon))$ so can $\widetilde{\gamma}(\mathsf{x})$. Note we can upper bound $\mathsf{poly}(d, \log Q, \log(1/\epsilon)) = \mathsf{poly}(d, \lambda)$ by a fixed polynomial since Q is a polynomial and ϵ is noticeable. Moreover, $\mathsf{F}_{\mathsf{Boy}}$ can be computed with $k \times L_d$ additions so we have $T_{\mathsf{F}} = L_d \cdot \mathsf{poly}(\log Q, \log(1/\epsilon))$. Similarly this is upper bounded by $L_d \cdot \mathsf{poly}(\lambda)$ for some fixed polynomial as desired.

□

5.5.1 Constructing d-wise Linearly Independent Hash Function

Here, we show an explicit construction of d-wise linearly independent hash function $h_{d\text{-wise}} : \{0,1\}^\ell \to \{0,1\}^{L_d}$ over $\mathbb{Z}_q$ for arbitrary integers d and ℓ and a prime q. We set $L_d = dt\lceil \log q \rceil$, where t is the smallest integer such that $q^t \geq 2^\ell$ and thus $L_d \leq 4d\ell$. To construct such a hash, we consider an arbitrary injective map $\iota : \{0,1\}^\ell \to \mathbb{F}_{q^t}$, where $\mathbb{F}_{q^t}$ is a finite field of size q^t. We also consider a natural bijection between $\mathbb{F}_{q^t}$ and $\mathbb{Z}_q^t$ specified by maps $\pi : \mathbb{F}_{q^t} \to \mathbb{Z}_q^t$ and $\pi^{-1} : \mathbb{Z}_q^t \to \mathbb{F}_{q^t}$, where both π and π^{-1} are additively homomorphic. For an integer n, we consider a map $\mathbf{G}_n^{-1} : \mathbb{Z}_q^n \to \{0,1\}^{n\lceil \log q \rceil}$ that maps a vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ to its binary representation in $\{0,1\}^{n\lceil \log q \rceil}$. Note that we have $\mathbf{G}_n^{-1}(\mathbf{a}) \cdot \mathbf{G}_n^\top = \mathbf{a}$ for any $\mathbf{a} \in \mathbb{Z}_q^n$, where $\mathbf{G}_n = \mathbf{I}_n \otimes (1, 2, \ldots, 2^{\lceil \log q \rceil})$ and we treat the binary string $\mathbf{G}_n^{-1}(\mathbf{a})$ as a row vector here. In this setting, we have the following lemma.

Lemma 4. *The function $h_{d\text{-wise}} : \{0,1\}^\ell \to \{0,1\}^{L_d}$ defined as*

$$h_{d\text{-wise}}(\mathsf{x}) = \mathbf{G}_{td}^{-1}\Big(\pi\big(\iota(1)\big), \pi\big(\iota(\mathsf{x})\big), \pi\big(\iota(\mathsf{x})^2\big), \ldots, \pi\big(\iota(\mathsf{x})^{d-1}\big)\Big)$$

is d-wise linearly independent over $\mathbb{Z}_q$.

Proof. For the sake of contradiction, let us assume that there exist mutually distinct $\mathsf{x}_1, \ldots, \mathsf{x}_d \in \{0,1\}^\ell$ such that $h_{d\text{-wise}}(\mathsf{x}_1), \ldots, h_{d\text{-wise}}(\mathsf{x}_d)$ are linearly dependent over $\mathbb{Z}_q$. Then, there exists a vector $\mathbf{v} = (v_1, \ldots, v_d)^\top \in \mathbb{Z}_q^d \backslash \{\mathbf{0}\}$ such that $\sum_{i=1}^d v_i h_{d\text{-wise}}(\mathsf{x}_i) = \mathbf{0}$. We then have $\sum_{i=1}^d v_i h_{d\text{-wise}}(\mathsf{x}_i) \cdot \mathbf{G}_{td}^\top = \sum_{i=1}^d v_i h'(\mathsf{x}_i) = \mathbf{0}$, where $h'(\mathsf{x}_i) = (\pi(\iota(1)), \pi(\iota(\mathsf{x}_i)), \ldots, \pi(\iota(\mathsf{x}_i)^{d-1}))$. Since π is an additively homomorphic and injective map, this implies $\sum_{i=1}^d v_i h''(\mathsf{x}_i) = \mathbf{0}$, where $h''(\mathsf{x}_i) = (\iota(1), \iota(\mathsf{x}_i), \ldots \iota(\mathsf{x}_i)^{d-1}) \in (\mathbb{F}_{q^t})^d$. However, this contradicts the fact that $(h''(\mathsf{x}_i))_{i\in[d]}$ are linearly independent over $\mathbb{F}_{q^t}$ (and thus over $\mathbb{Z}_q$), since these vectors constitute Vandermonde matrix with $(\iota(\mathsf{x}_i))_{i\in[d]}$ being mutually distinct. □

5.6 Partitioning Function Based on Substring Matching

While $\mathsf{F}_{\mathsf{Wat}}$ and $\mathsf{F}_{\mathsf{ParWat}}$ both achieve a large $\gamma_{\mathsf{min}} = O(\epsilon^{1/d}/Q)$ for $d = 2$ or even larger d and covers both the pairing groups and lattice settings, respectively, one caveat is that the size it takes to describe the partitioning function (i.e., partitioning key K) is large. It is often the case that when using partitioning function with cryptographic primitives, we need to secretly compute

$\mathsf{F}(K,\cdot)$, in which case we must embed K into the system parameters. Therefore, having smaller description size for K often ends up with smaller system parameter and is desirable.

In this section, we revisit the partitioning functions based on substring matching that appear in [Lys02, BB04b, CHKP10, FHPS13, Bit17, Koh19]. While the partitioning function is more complex compared to $\mathsf{F}_{\mathsf{Wat}}$ and $\mathsf{F}_{\mathsf{ParWat}}$, it offers a much smaller description size.

Let $\ell := \ell(\lambda)$, $n := n(\lambda)$, and $\eta := \eta(\lambda)$ be integers of polynomial size and $\Sigma := \Sigma_\lambda$ be an alphabet. Here, we focus on the cases where $\Sigma = \{0,1\}$ and $\Sigma = \{1,2,\ldots,|\Sigma|\}$ for polynomially bounded $|\Sigma|$. We consider an encoding function

$$\mathsf{Encode} : \{0,1\}^\ell \to \Sigma^n.$$

Let us also define $\mathcal{K} := ([n] \times \Sigma)^{\leq \eta}$. Namely, a key $K \in \mathcal{K}$ is in the form of $K = \{(I_i, \sigma_i)\}_{i \in [\eta']}$, where we have $\eta' \leq \eta$ and $I_i \in [n]$ and $\sigma_i \in \Sigma$ for all $i \in [\eta']$. We then define the function $\mathsf{F}_{\mathsf{SSM}} : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}$ as

$$\mathsf{F}_{\mathsf{SSM}}(K,\mathsf{x}) = \begin{cases} 0 & \text{if} \quad \sigma_i = \mathsf{Encode}(\mathsf{x})_{I_i} \quad \forall i \in [\eta'] \\ 1 & \text{othrewise} \end{cases}, \tag{39}$$

where $\mathsf{Encode}(\mathsf{x})_{I_i}$ is the I_i-th symbol of the string $\mathsf{Encode}(\mathsf{x}) \in \Sigma^n$. Previously works required Encode to be an error correcting code with large enough minimal distance.[11][12] In this work, we require the following stronger property for Encode.

Definition 10 (Small triple overlap property)**.** *We say that an encoding function* $\mathsf{Encode} : \{0,1\}^\ell \to \Sigma^n$ *has small triple overlap property with parameter* $c := c(\lambda)$ *if the following properties hold:*

- *For arbitrary* $\mathsf{x}_1, \mathsf{x}_2 \in \{0,1\}^\ell$ *with* $\mathsf{x}_1 \neq \mathsf{x}_2$*, we have*

$$\#\{i \in [n] : \mathsf{Encode}(\mathsf{x}_1)_i = \mathsf{Encode}(\mathsf{x}_2)_i\} \leq (1-c)n,$$

  where $\mathsf{Encode}(\mathsf{x}_b)_i$ *for* $b \in \{1,2\}$ *denotes the* i*-th symbol of the codeword* $\mathsf{Encode}(\mathsf{x}_b) \in \Sigma^n$*.*

- *For arbitrary but mutually distinct* $\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3 \in \{0,1\}^\ell$*, we have*

$$\#\{i \in [n] : \mathsf{Encode}(\mathsf{x}_1)_i = \mathsf{Encode}(\mathsf{x}_2)_i = \mathsf{Encode}(\mathsf{x}_3)_i\} \leq (1-c)^2 n,$$

  where $\mathsf{Encode}(\mathsf{x}_b)_i$ *for* $b \in \{1,2,3\}$ *denotes the* i*-th symbol of the codeword* $\mathsf{Encode}(\mathsf{x}_b) \in \Sigma^n$*.*

As we will soon see in Theorem 5, $\mathsf{F}_{\mathsf{SSM}}$ instantiated with an encoding function Encode satisfying Def. 10 is a partitioning function that admits fine-tuned approximation, which leads to better reduction costs for many VRFs and IBEs. Unfortunately, we do not know an explicit construction of such encoding algorithm, where an explicit construction refers to an efficient deterministic algorithm that takes only ℓ, n, Σ, and x as input and outputs $\mathsf{Encode}(\mathsf{x})$. However, as we show in the following lemma, random 3-wise independent hash functions with appropriately chosen parameters satisfy the above properties except for an exponentially small probability. We therefore can pick a random 3-wise independent hash and use it as a description of an encoding function satisfying Def. 10.

[11] Both in previous works and our work, we do not need efficient decoding algorithm for Encode.

[12] Many previous works [BB04b, CHKP10, Yam17] call the encoding function that admits a partitioning function based on sub-string matching "admissible hash".

Lemma 5. *Let us consider a family of 3-wise independent hash $\mathcal{H}_{\ell,\Sigma,n} = \{h : \{0,1\}^\ell \to \Sigma^n\}$. Then, randomly chosen h from $\mathcal{H}_{\ell,\Sigma,n}$ satisfies small triple overlap property as per Def. 10 with parameter $c < 1 - 1/|\Sigma|$ except for probability*

$$p_{c,\ell,\Sigma,n} := 2^{2\ell+1} \exp\left(-2\left(1 - c - \frac{1}{|\Sigma|}\right)^2 n\right) + 2^{3\ell+1} \exp\left(-2\left((1-c)^2 - \frac{1}{|\Sigma|^2}\right)^2 n\right).$$

In particular, under the following settings, h chosen from $\mathcal{H}_{\ell,\Sigma,n}$ satisfies small triple overlap probability with probability more than $1 - 2^{-\ell}$.

Binary alphabets. *$\Sigma = \{0,1\}$, c is a constant with $c < 1/2$, $n = 4\ell/((1-c)^2 - 1/4) = O(\ell)$.*

Polynomial alphabets. *$\Sigma = \{1, 2, \ldots, 2\ell^\nu\}$, $c = 1 - 1/\ell^\nu$, where ν is a constant with $1 > \nu > 0$, and $n = 3\ell^{1+4\nu} = O(\ell^{1+4\nu})$.*

Proof. The latter part of the lemma follows from the former part. We therefore focus on the former part. We first bound the probability that randomly chosen h does not satisfy the second property of Def. 10. Let us fix mutually distinct x_1, x_2, and x_3 in $\{0,1\}^\ell$. For each $i \in \Sigma$, let E_i be the event that $h(\mathsf{x}_1)_i = h(\mathsf{x}_2)_i = h(\mathsf{x}_3)_i$ holds, where the probability is taken over the choice of $h \xleftarrow{\$} \mathcal{H}_{\ell,\Sigma,n}$. Since $\mathcal{H}_{\ell,\Sigma,n}$ is a family of 3-wise independent hash, $(h(\mathsf{x}_1), h(\mathsf{x}_2), h(\mathsf{x}_3))$ is distributed uniformly at random over Σ^3. In particular, we have that $\mathsf{E}_1, \ldots, \mathsf{E}_n$ are independent and $\Pr[\mathsf{E}_i] = 1/|\Sigma|^2$. Using Hoeffding's bound, we have $\Pr[\sum_{i=1}^n \mathsf{E}_i \geq (1-c)^2 n] \leq 2\exp(-2((1-c)^2 n - n/|\Sigma|^2)^2/n) = 2\exp(-2((1-c)^2 - 1/|\Sigma|^2)^2 n)$, where we abuse the notation here and denote by E_i the random variable that takes the value 1 when E_i occurs and 0 otherwise. Noticing that $\sum_{i=1}^n \mathsf{E}_i \geq (1-c)^2 n \Leftrightarrow \#\{j : \mathsf{Encode}(\mathsf{x}_1)_j = \mathsf{Encode}(\mathsf{x}_2)_j = \mathsf{Encode}(\mathsf{x}_3)_j\} \geq (1-c)^2 n$ and taking union bound over all possible x_1, x_2, and $\mathsf{x}_3 \in \{0,1\}^\ell$, we conclude that randomly chosen 3-wise independent hash satisfies the second property of Def. 10 except for probability $2^{3\ell+1}\exp(-2((1-c)^2 - 1/|\Sigma|^2)^2 n)$.

We then consider the probability that randomly chosen h does not satisfy the first property of Def. 10. By the similar argument to the above, for any distinct x_1 and x_2, we can bound the probability that the number of position i for which $h(\mathsf{x}_1)_i = h(\mathsf{x}_2)_i$ exceeds $(1-c)n$ by $2\exp(-2(1 - c - 1/|\Sigma|)^2 n)$. Then, by taking the union bound over all possible x_1 and x_2, we can bound the probability by $2^{2\ell+1}\exp(-2(1-c-1/|\Sigma|)^2 n)$.

Finally, by taking the union bound again, we can conclude that randomly chosen h satisfies both properties of Def. 10 except for probability $2^{2\ell+1}\exp(-2(1-c-1/|\Sigma|)^2 n) + 2^{3\ell+1}\exp(-2((1-c)^2 - 1/|\Sigma|^2)^2 n)$ as desired. □

Remark 3 (On the usage of 3-wise independent hash functions)**.** *We remark that to construct an error correcting code, more standard approach would be to choose random matrix $\mathbf{A}$ over $\mathbb{Z}_{|\Sigma|}^{n \times \ell}$ and define h as $h(\mathsf{x}) := \mathbf{A}\mathsf{x}^\top$, where $\mathsf{x} \in \{0,1\}^\ell$ is treated as a row vector (See e.g., [Gol08]). We choose to use 3-wise independent hash function instead, since the description size for the encoding function is much shorter. Looking ahead, having shorter description size for the function leads to shorter system parameters when we consider applications to IBEs and VRFs, since we need to include the function description into the system parameters in these applications.*

We then show the following theorem.

Theorem 5. *Let us assume that $\mathsf{Encode} : \{0,1\}^\ell \to \Sigma^n$ has small triple overlap property with parameter $c = c(\lambda)$ as per Defnition 10. Then, for a real number $\epsilon = \epsilon(\lambda)$ in $(0, 1/2]$ and a*

positive integer $Q = Q(\lambda)$, $\mathsf{F}_{\mathsf{SSM}}$ defined as in Eq. (39) is a $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$-partitioning function such that

$$\eta = \frac{\omega(\log(\lambda))}{\log(1/1-c)}, \quad \gamma_{\mathsf{min}} = \frac{1}{2|\Sigma|^{\eta'}}, \quad T_{\mathsf{F}} = \mathsf{poly}(\lambda, n), \text{ and } T_{\mathsf{approx}} = Q \cdot \mathsf{poly}(\lambda, n),$$

where

$$\eta' = \left\lceil \frac{\log(2Q/\sqrt{\epsilon})}{\log(1/1-c)} \right\rceil,$$

and $\mathsf{poly}(\lambda, n)$ is a fixed polynomial independent from Q and ϵ.

Proof. We define $\mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$ as follows:

$\mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$: It sets $\eta' := \left\lceil \frac{\log(2Q/\sqrt{\epsilon})}{\log(1/1-c)} \right\rceil$ and samples random subset $I = \{I_1, \ldots, I_{\eta'}\} \subseteq [n]$ and random symbol $\sigma_i \xleftarrow{\$} \Sigma$ for $i \in [\eta']$. It then outputs $K = \{(I_i, \sigma_i)\}_{i \in [\eta']}$.

To prove this lemma, we need to show that PrtSmp satisfies three properties in Def. 9.

First property. We show that K output by the above algorithm is always in $\mathcal{K}_\lambda$ for large enough λ. Since K output by $\mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$ is in $([n] \times \Sigma)^{\eta'}$ and $\mathcal{K} = ([n] \times \Sigma)^{\eta}$, it suffices to show that $\eta'(\lambda) < \eta(\lambda)$ holds for large enough λ. This holds since we have

$$\eta' = \left\lceil \frac{\log(2Q/\sqrt{\epsilon})}{\log(1/1-c)} \right\rceil = \left\lceil \frac{\log(\mathsf{poly}(\lambda))}{\log(1/1-c)} \right\rceil \leq \frac{\omega(\log(\lambda))}{\log(1/1-c)},$$

where the second equality follows from $Q(\lambda) = \mathsf{poly}(\lambda)$ and $1/\epsilon(\lambda) = \mathsf{poly}(\lambda)$ and the last inequality holds for large enough λ.

Second property. For $\mathsf{x} \in \{0,1\}^\ell$, let $\mathsf{E}(\mathsf{x})$ be the event that $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 0$ holds. It is easy to see that

$$\Pr[\mathsf{E}(\mathsf{x})] = \Pr[\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i \ \forall i \in [\eta']] = \frac{1}{|\Sigma|^{\eta'}}$$

holds for all x, where $K \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$. We also observe that for x and x' with $\mathsf{x} \neq \mathsf{x}'$, we have

$$\begin{aligned}
\Pr[\mathsf{E}(\mathsf{x}) \wedge \mathsf{E}(\mathsf{x}')] &= \Pr\left[\left(\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i \ \forall i \in [\eta']\right) \wedge I \subseteq \underbrace{\{j : \mathsf{Encode}(\mathsf{x})_j = \mathsf{Encode}(\mathsf{x}')_j\}}_{:=J} \right] \\
&= \Pr\left[\left(\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i \ \forall i \in [\eta']\right) \ \middle| \ I \subseteq J \} \right] \cdot \Pr[I \subseteq J\}] \\
&= \frac{1}{|\Sigma|^{\eta'}} \cdot \prod_{i=0}^{\eta'-1} \left(\frac{\#J - i}{n - i} \right) \qquad (40) \\
&\leq \frac{1}{|\Sigma|^{\eta'}} \cdot \prod_{i=0}^{\eta'-1} \left(\frac{(1-c)n - i}{n - i} \right) \\
&\leq \left(\frac{1-c}{|\Sigma|} \right)^{\eta'} \qquad (41)
\end{aligned}$$

where the probability is taken over the choice of $K \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q, \epsilon)$. The third equation above follows from the fact that for any fixed I, the event $\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i$ happens with probability

$1/|\Sigma|$ independently for each $i \in [\eta']$ and the first inequality follows from the fact that $\mathsf{Encode}(\mathsf{x})$ and $\mathsf{Encode}(\mathsf{x}')$ differ in at least cn positions by the first property of Def. 10.

For $\vec{\mathsf{x}} = (\mathsf{x}^*, \mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(Q)})$, we then define $\widetilde{\gamma}(\vec{\mathsf{x}})$ as

$$\widetilde{\gamma}(\vec{\mathsf{x}}) := \Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j \in [Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})].$$

By the same analysis as the proof of Lemma 2, we have

$$\gamma(\vec{\mathsf{x}}) \geq \underbrace{\Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j \in [Q]} \Pr\left[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})\right]}_{=\widetilde{\gamma}(\vec{\mathsf{x}})} \geq \frac{1 - Q(1-c)^{\eta'}}{|\Sigma|^{\eta'}} \geq \frac{1 - \sqrt{\epsilon}/2}{|\Sigma|^{\eta'}} \geq \underbrace{\frac{1}{2|\Sigma|^{\eta'}}}_{=\gamma_{\mathsf{min}}(\vec{\mathsf{x}})}$$

where the second inequality follows from Eq. (41) and the third inequality follows from $(1-c)^{\eta'} \leq \sqrt{\epsilon}/2Q$, which holds by our choice of η'. We therefore have proven the first two inequalities of Eq. (25). It remains to prove the third inequality. To do so, we first observe that for mutually distinct x, x', and x'', we have

$$\begin{aligned}
& \Pr[\mathsf{E}(\mathsf{x}) \wedge \mathsf{E}(\mathsf{x}') \wedge \mathsf{E}(\mathsf{x}'')] \\
= \; & \Pr\left[\left(\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i \; \forall i \in [\eta']\right) \wedge I \subseteq \{j : \mathsf{Encode}(\mathsf{x})_j = \mathsf{Encode}(\mathsf{x}')_j = \mathsf{Encode}(\mathsf{x}'')_j\}\right] \\
= \; & \Pr\left[\left(\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i \; \forall i \in [\eta']\right) \;\middle|\; I \subseteq \{j : \mathsf{Encode}(\mathsf{x})_j = \mathsf{Encode}(\mathsf{x}')_j = \mathsf{Encode}(\mathsf{x}'')_j\}\right] \\
& \cdot \Pr[I \subseteq \{j : \mathsf{Encode}(\mathsf{x})_j = \mathsf{Encode}(\mathsf{x}')_j = \mathsf{Encode}(\mathsf{x}'')_j\}] \\
\leq \; & \frac{1}{|\Sigma|^{\eta'}} \cdot \prod_{i=0}^{\eta'-1} \left(\frac{(1-c)^2 n - i}{n - i}\right) \\
\leq \; & \left(\frac{(1-c)^2}{|\Sigma|}\right)^{\eta'}, \qquad (42)
\end{aligned}$$

where the first inequality above follows from the fact that for any fixed I, the event $\mathsf{Encode}(\mathsf{x})_{I_i} = \sigma_i$ happens with probability $1/|\Sigma|$ independently for each $i \in [\eta']$ and the number of indices j such that $\mathsf{Encode}(x)_j = \mathsf{Encode}(x)'_j = \mathsf{Encode}(x)''_j$ is at most $(1-c)^2 n$ due to the small triple overlap property (Def. 10).

By the same analysis as the proof of Lemma 2, we have

$$|\gamma(\vec{\mathsf{ID}}) - \widetilde{\gamma}(\vec{\mathsf{ID}})| \leq \sum_{1 \leq j < k \leq Q} \Pr\left[\mathsf{E}(\mathsf{ID}^*) \wedge \mathsf{E}(\mathsf{ID}^{(j)}) \wedge \mathsf{E}(\mathsf{ID}^{(k)})\right]. \qquad (43)$$

We then have

$$\text{Eq. (43)} \leq \frac{Q^2}{2} \cdot \left(\frac{(1-c)^2}{|\Sigma|}\right)^{\eta'} \leq \frac{\epsilon}{8|\Sigma|^{\eta'}} = \frac{\gamma_{\mathsf{min}}\epsilon}{4} < \frac{\gamma_{\mathsf{min}}\epsilon}{3},$$

where the first inequality follows from Eq. (42) and the second inequality follows from $(1-c)^{\eta'} \leq \sqrt{\epsilon}/2Q$, which holds by our choice of η'. This completes the third inequality of Eq. (25).

Third property. Finally, we show the third property of Def. 9. We first observe that to compute $\widetilde{\gamma}(\vec{\mathsf{x}}) = \Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j \in [Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$, it suffice to bound the time required for computing $\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$ for each of $j \in [Q]$, since $\Pr[\mathsf{E}(\mathsf{x}^*)] = 1/|\Sigma|^{\eta'}$ can be computed directly. By Eq. (40), we have $|\Sigma|^{\eta'} \cdot \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] = \prod_{i=1}^{\eta'} \left(\frac{\#J_{\mathsf{x}^*,\mathsf{x}} - i}{n - i}\right)$, where $J_{\mathsf{x}^*,\mathsf{x}} = \{j : \mathsf{Encode}(\mathsf{x})_j =$

$\mathsf{Encode}(\mathsf{x}^*)_j\}$. These quantities can be computed by in time $\mathsf{poly}(\eta', n, \log|\Sigma|) \leq \mathsf{poly}(\lambda, n)$ for some fixed polynomial, where the inequality follows from the fact that $\log|\Sigma|$ and η' are $O(\log\lambda)$. Moreover, it is straightforward to see that $\mathsf{F}_{\mathsf{SSM}}$ can be computed in time $\mathsf{poly}(\eta', n, \log|\Sigma|) \leq \mathsf{poly}(\lambda, n)$ for some fixed polynomial as desired. □

As a corollary of Lemma 5 and Theorem 5, we obtain the following theorem.

Theorem 6. *For any integer function $\ell := \ell(\lambda)$ and constants $\mu > 1$ and $0 < \nu < 1$, there exist a function $n = n(\lambda)$ and a family of efficient and efficiently samplable hash functions $\mathcal{H}_\lambda : \{0,1\}^\ell \to \Sigma^n$ such that $\mathsf{F}_{\mathsf{SSM}}$ defined as in Eq. (39) by setting $\mathsf{Encode} := h$ for $h \xleftarrow{\$} \mathcal{H}_\lambda$ is a $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$-partitioning function except for probability $2^{-\ell}$ under the following parameter settings:*

(Binary alphabets) *In binary alphabets setting, we have $\Sigma = \{0,1\}$ and*

$$n = \Theta(\ell), \quad \eta = \omega(\log(\lambda)), \quad \gamma_{\mathsf{min}} = \frac{1}{4}\left(\frac{\sqrt{\epsilon}}{2Q}\right)^{\mu}, \quad T_{\mathsf{F}} = \mathsf{poly}(\lambda, \ell), \quad T_{\mathsf{approx}} = Q \cdot \mathsf{poly}(\lambda, \ell),$$

where $\mathsf{poly}(\lambda, \ell)$ is some fixed polynomial independent from Q and ϵ.

(Polynomial alphabets) *In polynomial alphabets setting, we have good property $\Sigma = \{1, 2, \ldots, 2\ell^\nu\}$ and*

$$n = \Theta(\ell^{1+4\nu}), \quad \eta = \omega(1), \quad \gamma_{\mathsf{min}} = \frac{\sqrt{\epsilon}}{\omega(1) \cdot \ell^\nu Q}, \quad T_{\mathsf{F}} = \mathsf{poly}(\lambda, \ell), \quad T_{\mathsf{approx}} = Q \cdot \mathsf{poly}(\lambda, \ell),$$

where $\omega(1)$ can be any function that grows faster than 1 asymptotically (e.g., $\log\log(\lambda)$) and $\mathsf{poly}(\lambda, \ell)$ is some fixed polynomial independent from Q and ϵ.

The description of h requires $3n\lceil\log|\Sigma|\rceil$ bits for both cases.

Proof. The proof is obtained by combining Lemma 5 and Theorem 5 straightforwardly, by noting that a 3-wise independent hash function $h : \{0,1\}^\ell \to \Sigma^n$ can be represented using $a_0, a_1, a_2 \in \mathbb{F}_{2^k}$, where $h(x) = a_0 + a_1 x + a_2 x^2$, and both the input and output domains are embedded within a finite field $\mathbb{F}_{2^k}$ of size 2^k, where $k = n\lceil\log|\Sigma|\rceil$, in a natural manner. □

Remark 4 (Tradeoffs provided by setting ν)**.** *We want η and n to be as small as possible, since as they get smaller, we typically are able to obtain VRF/IBE schemes with better space efficiency (e.g., [Yam17, Kat17, Koh19]). Similarly, we want γ_{min} to be as large as possible, since the reduction costs of the schemes become tighter as it gets larger. By setting μ (resp., ν) close to 1 (resp., 0) in binary alphabet case (resp., polynomial alphabet case), these requirements can be satisfied at the same time asymptotically. However, choosing ν and μ in a way that leads to better asymptotic parameters may result in worse concrete space efficiency/reduction cost due to larger hidden constant terms when we consider concrete parameters.*

Remark 5 (Comparison with previous works)**.** *Here, we compare our bound on γ_{min} with that shown in previous works [Jag15, Koh19]. For simplicity, we ignore poly-logarithmic factors here. In binary alphabet case, we achieve $\gamma_{\mathsf{min}} = (\sqrt{\epsilon}/Q)^\mu$ for arbitrary constant $\mu > 1$, which improves $\gamma_{\mathsf{min}} = (\epsilon/Q)^\mu$ shown by Jager [Jag15]. The improvement is due to our fine-tuned analysis that uses small triple overlap property of the underlying encoding function. In the polynomial-size alphabet case, we achieve $\sqrt{\epsilon}/\ell^\nu Q$ for arbitrary $1 \geq \nu > 0$, whereas Kohl [Koh19] showed*

$(\epsilon/\ell Q)^{1+1/\nu}$.[13] *The reason why our bound is better is twofold. Firstly, we use an error-correcting code whose gap between the quantities* $1-c$ *and* $1/|\Sigma|$ *is quite narrow. This choice is pivotal, because as we can observe from the statement of Theorem 5, as the gap between the quantities* $1-c$ *and* $1/|\Sigma|$ *widens, where* c *represents the relative distance of the code,* γ_{min} *becomes smaller. In our setting, we have* $1-c \approx \ell^{-\nu} \approx 1/|\Sigma|$*, while she relies on Reed-Solomon code and has* $1-c \approx \ell^{-\nu}$ *and* $1/|\Sigma| \approx \ell^{-1-\nu}$*. Secondly, we employ our fine-tuned analysis using the small triple overlap property here again. This leads to a further improvement on the bound by a factor of* $\sqrt{\epsilon}$*.*

6 Application to IBEs

Recall that the notion of the partitioning function [Yam17] abstracts out the core statistical properties useful for proving security of various cryptographic primitives. In Sec. 4, we essentially showed that if the underlying partitioning function admits good enough approximation for the quantity γ, then we can achieve better reduction costs in various security proofs than those obtained by existing techniques [Wat05, BR09]. Then, in Sec. 5, we showed that new and existing partitioning functions indeed admit good enough approximations. These arguments are divorced from the underlying cryptographic primitives and algebraic structures. In this section, we apply the tools we developed in Sec. 4 and 5 to the specific context of IBE. This allows us to prove improved reduction costs for Waters IBE [Wat05] and Agrawal-Boneh-Boyen IBE [ABB10a] and also yields a new scheme with good reduction costs. To formally prove these results in a unified manner, we show a template of the security proof for IBE that uses partitioning functions. We then prove the security of the respective IBE schemes using the template.

6.1 Security Proof Template for IBE

We show a security proof template for IBE schemes using the artificial abort paradigm.

Definition 11 (Partitioning-Based Reduction for IBE)**.** *We say that there is a* $(\epsilon_{\mathsf{S}}, \epsilon_{\mathsf{K}}, \epsilon_{\mathsf{E}}, \epsilon_{\mathsf{R}})$*-partioning-based reduction for an IBE scheme* $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *from a decision problem* $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ *with respect to a* $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$*-partitioning function with approximation* $\mathsf{F} = \{\mathsf{F} : \mathcal{K} \times \{0,1\}^{\ell} \to \{0,1\}\}$ *if there exists a tuple of efficient algorithms* $(\mathsf{SimSetup}, \mathsf{SimKeyGen}, \mathsf{SimEncrypt})$ *with the following syntax.*

$\mathsf{SimSetup}(K, \psi) \to (\mathsf{mpk}, \mathsf{td})$*. It takes as input a partitioning key* $K \in \mathcal{K}$ *and the problem instance* ψ *(output by either* $\mathcal{D}_0$ *or* $\mathcal{D}_1$*) and outputs a master public key* mpk *and a trapdoor* td*.*

$\mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$*. It takes as input a trapdoor* td *and an identity* $\mathsf{ID} \in \{0,1\}^{\ell}$ *and outputs a secret key* $\mathsf{sk}_{\mathsf{ID}}$*.*

$\mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}, \mathsf{M}) \to \mathsf{ct}$*. It takes as input a trapdoor* td*, an identity* $\mathsf{ID} \in \{0,1\}^{\ell}$*, and a message* M *and outputs a ciphertext* ct*.*

For these algorithms, we require the following properties. For describing the properties, we introduce a couple of notations here. For a string mpk*, we define a set* S_{mpk} *as* $S_{\mathsf{mpk}} := \{\mathsf{msk} :$

[13] Actually, Kohl [Koh19] adopts the technique of Waters [Wat05] and gives lower bound for γ, rather than giving both lower and upper bounds as Bellare and Ristenpart [BR09]. This leads to looser reduction cost when we consider applications to IBEs and VRFs in typical parameter settings, since it requires artificial abort using Monte Carlo method. To be fair, we analyze her function employing the technique of Bellare and Ristenpart and then compare the obtained bound on γ_{min} with ours.

$(\mathsf{mpk}, \mathsf{msk}) \in \mathsf{Setup}(1^\lambda)\}$. *For the distribution* $\mathcal{D}_b$ *with* $b \in \{0,1\}$ *and* K, $\mathsf{SimSetup}(K, \mathcal{D}_b)$ *denotes the output distribution of* $\mathsf{SimSetup}(K, \psi)$*, where* ψ *is sampled as* $\psi \xleftarrow{\$} \mathcal{D}_b$*. We also denote by* $\mathsf{SimSetup}(K, \mathcal{D}_b)|_{\mathsf{mpk}}$ *the distribution of* td *output by* $\mathsf{SimSetup}(K, \mathcal{D}_b)$ *conditioned on the first output being* mpk*. If* $\mathsf{mpk} \notin \mathsf{SimSetup}(K, \mathcal{D}_b)$*,* $\mathsf{SimSetup}(K, \mathcal{D}_b)|_{\mathsf{mpk}}$ *outputs* $\perp$*.*

Master public key simulatability: *For all* $K \in \mathcal{K}$ *and all* $\psi \in \mathcal{D}_0 \cup \mathcal{D}_1$*, the marginal distribution of* mpk *output by* $\mathsf{SimSetup}(K, \psi)$ *is within* ϵ_{S} *statistical distance of the marginal distribution of* mpk *output by* $\mathsf{Setup}(1^\lambda)$*. Moreover, the runtime of* $\mathsf{SimSetup}$ *and* Setup *are within some polynomial factor* $\mathsf{poly}(\lambda)$*.*

Secret key simulatability: *For all* $K \in \mathcal{K}$*, all* $\psi \in \mathcal{D}_0 \cup \mathcal{D}_1$*, all* $(\mathsf{mpk}, \mathsf{td}) \in \mathsf{SimSetup}(K, \psi)$ *such that* $S_{\mathsf{mpk}} \neq \emptyset$*, all* $\mathsf{msk} \in S_{\mathsf{mpk}}$*, and all* $\mathsf{ID} \in \{0,1\}^\ell$ *such that* $\mathsf{F}(K, \mathsf{ID}) = 1$*, the following distributions are within* ϵ_{K} *statistical distance:*

$$\left\{\mathsf{sk}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID})\right\} \approx_{\epsilon_{\mathsf{K}}} \left\{\mathsf{sk}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})\right\}.$$

Moreover, the runtime of $\mathsf{SimKeyGen}$ *and* KeyGen *are within some polynomial factor* $\mathsf{poly}(\lambda)$*.*

Ciphertext simulatability: *For all* $K \in \mathcal{K}$*, all* mpk *such that there exists* td *satisfying* $(\mathsf{mpk}, \mathsf{td}) \in \mathsf{SimSetup}(K, \mathcal{D}_0)$*, all* $\mathsf{ID}^* \in \{0,1\}^\ell$ *such that* $\mathsf{F}(K, \mathsf{ID}^*) = 0$*, and all* $\mathsf{M} \in \mathcal{M}$ *the following distributions are within* ϵ_{E} *statistical distance:*

$$\left\{\mathsf{ct} \xleftarrow{\$} \mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}^*, \mathsf{M})\right\} \approx_{\epsilon_{\mathsf{E}}} \left\{\mathsf{ct} \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M})\right\},$$

where td *is sampled as* $\mathsf{td} \xleftarrow{\$} \mathsf{SimSetup}(K, \mathcal{D}_0)|_{\mathsf{mpk}}$*. Moreover, the runtime of* $\mathsf{SimEncrypt}$ *and* $\mathsf{Encrypt}$ *are within some polynomial factor* $\mathsf{poly}(\lambda)$*.*

Ciphertext randomizability: *For all* $K \in \mathcal{K}$*, all* mpk *such that there exists* td *satisfying* $(\mathsf{mpk}, \mathsf{td}) \in \mathsf{SimSetup}(K, \mathcal{D}_1)$*, all* $\mathsf{ID}^* \in \{0,1\}^\ell$ *such that* $\mathsf{F}(K, \mathsf{ID}^*) = 0$*, and all* $\mathsf{M}, \mathsf{M}^* \in \mathcal{M}$ *the following distributions are within* ϵ_{R} *statistical distance:*

$$\left\{\mathsf{ct} \xleftarrow{\$} \mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}^*, \mathsf{M})\right\} \approx_{\epsilon_{\mathsf{R}}} \left\{\mathsf{ct} \xleftarrow{\$} \mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}^*, \mathsf{M}^*)\right\},$$

where td *is sampled as* $\mathsf{td} \xleftarrow{\$} \mathsf{SimSetup}(K, \mathcal{D}_1)|_{\mathsf{mpk}}$*.*

The following establishes the security of an IBE scheme with a partitioning-based reduction. Below, for all of the constructions provided in this work, the dominant runtime overhead of the reduction is T_{approx} as $Q \cdot (T_{\mathsf{F}} + \mathsf{poly}(\lambda)) \lesssim t$.

Theorem 7. *Assume that there is a* $(\epsilon_{\mathsf{S}}, \epsilon_{\mathsf{K}}, \epsilon_{\mathsf{E}}, \epsilon_{\mathsf{R}})$*-partioning-based reduction for an IBE scheme* $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *from a decision problem* $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ *with respect to a* $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$*-partitioning function* F*. Then, if there is an* (t, Q, ϵ)*-adversary* A *against the* $\mathsf{IND\text{-}CPA}$ *security of the* IBE *scheme with a polynomial* Q *and non-negligible* ϵ*, there is an* (ϵ', t')*-adversary* A' *against the problem* $\mathcal{D}$ *such that*

$$t' = t + T_{\mathsf{approx}} + Q \cdot (T_{\mathsf{F}} + \mathsf{poly}(\lambda)), \qquad \epsilon' \geq \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon - (2\epsilon_{\mathsf{S}} + 2Q \cdot \epsilon_{\mathsf{K}} + \epsilon_{\mathsf{E}} + \epsilon_{\mathsf{R}}),$$

for a non-negligible ϵ' *and infinitely many* $\lambda \in \mathbb{N}$*. Moreover,* $\mathsf{poly}(\lambda)$ *is roughly the overhead incurred by running the simulated algorithms compared to the real* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt})$ *algorithms.*

Proof. We prove the theorem by a sequence of games. Let ϵ_i denote the advantage of A in Game_i. Below, we use the fact that since $\epsilon(\lambda)$ is non-negligible, there exists a noticeable function $\epsilon^*(\lambda)$ such that $\epsilon(\lambda) \geq \epsilon^*(\lambda)$ for infinitely many $\lambda \in \mathbb{N}$.

Game_0**:** This is the real IND-CPA game. By assumption, we have $\epsilon_0 = \epsilon$.

Game_1**:** In this game, we generate the partitioning key $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon^*\right)$ at the end of the game, independently from anything else. Even though we do not embed K into the parameters (i.e., K is information theoretically hidden from the adversary), we introduce the *artificial abort* step here. Note that due to our assumption and Def. 9, Item 1, for large enough λ, we have $K \in \mathcal{K}$ and the properties in Items 2 and 3 hold.

Concretely, let ID^* be the challenge identity, $\mathsf{ID}^{(i)}$ be the i-th ($i \in [Q]$) identity queried as part of the key-extraction query, $\vec{\mathsf{ID}} = (\mathsf{ID}^*, \mathsf{ID}^{(1)}, \cdots, \mathsf{ID}^{(Q)})$, coin the random bit sampled by the challenger, and $\widehat{\mathsf{coin}}$ the guess A outputs. At the end of the game, the challenger checks if the event $\mathsf{F}(K, \mathsf{ID}^{(1)}) = \cdots = \mathsf{F}(K, \mathsf{ID}^{(Q)}) = 1 \wedge \mathsf{F}(K, \mathsf{ID}^*) = 0$ occurs, and if not (denoted as event Bad), it ignores A's output and outputs a random guess $\mathsf{coin}' \xleftarrow{\$} \{0,1\}$ on behalf of A. From Def. 9, Item 2, event Bad occurs with probability $1 - \gamma(\vec{\mathsf{ID}})$. If event Bad does not occur, the challenger computes γ_{min} and $\widetilde{\gamma}(\vec{\mathsf{ID}})$, and outputs a random guess $\mathsf{coin}' \xleftarrow{\$} \{0,1\}$ on behalf of A with probability $1 - \gamma_{\mathsf{min}}/\widetilde{\gamma}(\vec{\mathsf{ID}})$ (denoted as event AAbort). If neither events Bad nor AAbort occur, the challenger uses A's guess $\mathsf{coin}' = \widehat{\mathsf{coin}}$.

Due to Def. 9, Item 3, the challenger's runtime overhead compared to Game_0 is $T_{\mathsf{approx}}(Q, \epsilon) + Q \cdot T_{\mathsf{F}}(Q, \epsilon)$. Due to Def. 9, Item 2, we have $|\gamma(\vec{\mathsf{x}}) - \widetilde{\gamma}(\vec{\mathsf{x}})| < \frac{\gamma_{\mathsf{min}}(\lambda)}{3} \cdot \epsilon^* \leq \frac{\gamma_{\mathsf{min}}(\lambda)}{3} \cdot \epsilon$ for infinitely many λ. Then, due to Theorem 1, we have

$$\epsilon_1 = \left| \Pr[\mathsf{coin}' = \mathsf{coin}] - \frac{1}{2} \right| \geq \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon \geq \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon^*,$$

for infinitely many $\lambda \in \mathbb{N}$.

Game_2**:** In this game, the partitioning key K is chosen at the beginning of the game and once the Bad event is satisfied (i.e., $\mathsf{F}(K, \mathsf{ID}^{(i)}) = 0$ for $i \in [Q]$ or $\mathsf{F}(K, \mathsf{ID}^*) = 1$ during the game), the challenger aborts without running the game until the end. This is only a conceptual change and we have $\epsilon_2 = \epsilon_1$.

Game_3**:** In this game, we use $(\mathsf{mpk}, \mathsf{td}) \xleftarrow{\$} \mathsf{SimSetup}(K, \psi)$ to obtain mpk, where $\psi \xleftarrow{\$} \mathcal{D}_0$. However, we do not use td for the simulation at this point. Rather, we inefficiently recover msk corresponding to mpk and then use the msk to run the game. In more detail, the challenger inefficiently checks whether $S_{\mathsf{mpk}} = \emptyset$ and aborts if so. Otherwise, the challenger chooses msk from the conditional distribution of msk output by $\mathsf{Setup}(1^\lambda)$ with mpk being fixed, which we denote by $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)|_{\mathsf{mpk}}$. We claim that the difference between Game_3 and Game_2 can be bounded by ϵ_{S}.

The claim can be proven by using the master public key simulatability by using the standard fact that the application of any randomized function f does not increase the statistical distance. Here, we consider the marginal distribution of mpk output by $\mathsf{Setup}(1^\lambda)$ and that output by $\mathsf{SimSetup}$. We then consider a function f that takes as input mpk, samples $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)|_{\mathsf{mpk}}$, and outputs $(\mathsf{mpk}, \mathsf{msk})$. If we start from the former (resp., latter) distribution and then apply the function f, the joint distribution of $(\mathsf{mpk}, \mathsf{msk})$ will be that of Game_2 (resp., Game_3). Hence, $|\epsilon_3 - \epsilon_2| \leq \epsilon_{\mathsf{S}}$.

Game_4: In this game, we generate the challenge ciphertext by $\mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}^*, \mathsf{M}_{\mathsf{coin}})$. Since the challenger only has to answer challenge query for ID^* such that $\mathsf{F}(K, \mathsf{ID}^*) = 0$ due to the changes introduced in the previous games and since the the distribution of td follows that of $\mathsf{SimSetup}(K, \mathcal{D})|_{\mathsf{mpk}}$ from the view of the adversary, we can use the ciphertext simulatability to conclude $|\epsilon_4 - \epsilon_3| \leq \epsilon_{\mathsf{E}}$.

Game_5: In this game, we answer key-extraction queries by $\mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID})$ instead of $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{ID})$. Since the challenger only has to answer key queries for ID such that $\mathsf{F}(K, \mathsf{ID}) = 1$ due to the changes introduced in the previous games, we can use the secret key simulatability to conclude that $|\epsilon_5 - \epsilon_4| \leq Q \cdot \epsilon_{\mathsf{K}}$. Here, the multiplicative factor of Q comes from the fact that we have to do the change Q times. Note that msk is no longer necessary for answering the key queries in this game.

Game_6: In this game, we stop checking whether $S_{\mathsf{mpk}} = \emptyset$ and no longer recover msk. Instead, the challenger answers any key-extraction query for ID such that $\mathsf{F}(K, \mathsf{ID}) = 1$ by running $\mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID})$. Note that the game is now efficient again.

We claim that the view of the adversary in this game only changes by ϵ_{S} from the previous game. To see this, we observe that these games differ only when $S_{\mathsf{mpk}} = \emptyset$. From the master public key simulatability, the probability of $S_{\mathsf{mpk}} = \emptyset$ happening when mpk is sampled from $\mathsf{SimSetup}(K, \psi)$ can be bounded by ϵ_{S}, since otherwise we can construct an (inefficient) distinguisher that breaks the master public key simulatability by checking whether $S_{\mathsf{mpk}} = \emptyset$ or not. Hence, $|\epsilon_6 - \epsilon_5| \leq \epsilon_{\mathsf{S}}$.

Game_7: In this game, we sample ψ from $\mathcal{D}_1$ instead of $\mathcal{D}_0$. If there is an adversary who can distinguish this game from the previous one, we can construct a distinguisher A' against $\mathcal{D}$ such that $\mathsf{Adv}^{\mathcal{D}}(\mathsf{A}') = |\epsilon_7 - \epsilon_6|$. Note that such a distinguisher A' is efficient due to the modification we made in Game_6.

Below, we would like to invoke ciphertext randomizability to change the challenge ciphertext to random. However, we cannot yet invoke it since some information of the trapdoor td may be leaking from the secret keys $\mathsf{sk}_{\mathsf{ID}}$. Below, we undo the modifications so that td is only used to generate the challenge ciphertext.

Game_8: In this game, we again check whether $S_{\mathsf{mpk}} = \emptyset$ and abort if so. Otherwise, it is the same as the previous game, where note that we do not use $\mathsf{msk} \in S_{\mathsf{mpk}}$. Following the same argument as in Game_6, we have $|\epsilon_8 - \epsilon_7| \leq \epsilon_{\mathsf{S}}$ due to the master public key simulatability.

Game_9: In this game, we answer key-extraction queries by $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{ID})$ instead of $\mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID})$. Similarly to Game_5, we can use the secret key simulatability to conclude that $|\epsilon_9 - \epsilon_8| \leq Q \cdot \epsilon_{\mathsf{K}}$.

Game_{10}: Finally, in the last game, we choose a random message $\mathsf{M}^* \xleftarrow{\$} \mathcal{M}$ and encrypt it for generating the challenge ciphertext regardless of the value of coin. By the ciphertext randomizability, we have $|\epsilon_{10} - \epsilon_9| \leq \epsilon_{\mathsf{R}}$.

In Game_{10}, coin is information theoretically hidden from A, and hence, $\epsilon_{10} = 0$. Collecting all the bounds, we arrive at the theorem statement. □

6.2 Application to Waters IBE

Here, we apply our framework to Waters IBE [Wat05]. His IBE achieves the unique property of having short ciphertext consisting only of 2 group elements and security under the standard DBDH assumption or even under the CBDH assumption if we slightly modify it using Goldreich-Levin's hardcore bit function [GL89] as we discuss in App. A.4 (See also [KY16]). For Waters IBE, we improve the reduction cost from $O(\epsilon^2/Q\ell)$ to $O(\epsilon^{1.5}/Q\ell)$, where by reduction cost we mean the advantage of the DBDH solving algorithm obtained by a (t, Q, ϵ)-adversary against the IBE. Here, we ignore the difference between the running time of the DBDH solving algorithms, since they are $t + Q \cdot \mathsf{poly}(\lambda)$ in both cases and their difference can be ignored in most of the interesting parameters settings. More formally, we obtain the following theorem:

Theorem 8. *If there is an $(t_{\mathsf{A}}, Q, \epsilon_{\mathsf{A}})$-adversary A against the $\mathsf{IND\text{-}CPA}$ security of the Waters IBE scheme, there is an adversary B that breaks the DBDH problem with advantage ϵ_{B} and t_{B} such that*

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}^{1.5}}{21Q\ell}, \quad t_{\mathsf{B}} = t_{\mathsf{A}} + O(Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda) \tag{44}$$

where $Q \leq p\sqrt{\epsilon_{\mathsf{A}}}/\ell\sqrt{3}$ and $\mathsf{poly}(\lambda)$ is roughly the overhead incurred by the running the simulated algorithms compared to the real $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt})$ algorithms.

The proof of the theorem can be obtained by observing that the original proof of Waters IBE follows the template of partitioning-based reduction for IBE in Def. 11 and plugging in our analysis on $\mathsf{F}_{\mathsf{Wat}}$ in Theorem 2 into our template. In App. A, we provide the proof of the theorem and necessary background, including the description of the Waters IBE scheme and partitioning-based reduction for the scheme.

6.3 Applications to ABB IBE and Its Variant

Here, we apply our framework to ABB IBE [ABB10a], which is one of the most important lattice IBE schemes, since it achieves the shortest ciphertext size and computational efficiency among the existing schemes. Conventionally, the reduction cost for ABB IBE was considered to be $O(\epsilon^2/qQ)$, employing the partitioning strategy based on $\mathsf{F}_{\mathsf{Boy}}$. However, as we note in Remark 2, our formal analysis reveals that they are only lower bounded by $O(\epsilon^3/Q^2)$, which is much worse. Using our new analysis on $\mathsf{F}_{\mathsf{Boy}}$, we can improve it to be $O(\epsilon^2/Q^2)$. Furthermore, by using our analysis on new partitioning function $\mathsf{F}_{\mathsf{ParWat}}$ with $d = 3$, this can be further improved to be $O(\epsilon^{1.5}/qQ)$. More formally, we obtain the following theorem:

Theorem 9. *If there is an $(t_{\mathsf{A}}, Q, \epsilon_{\mathsf{A}})$-adversary A against the $\mathsf{IND\text{-}CPA}$ security of the ABB IBE scheme, there is an adversary B that breaks the LWE problem with advantage ϵ_{B} and t_{B} such that*

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}^{1.5}}{12qQ} - \mathsf{negl}(\lambda), \quad t_{\mathsf{B}} = t_{\mathsf{A}} + Q \cdot \mathsf{poly}(\lambda) \tag{45}$$

where $q^n \geq 2 \cdot Q/\sqrt{\epsilon_{\mathsf{A}}}$ holds for dimension n of the scheme and $\mathsf{poly}(\lambda)$ is roughly the overhead incurred by the running the simulated algorithms compared to the real $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt})$ algorithms.

We also consider a variant of ABB IBE, where we hash an identity using d-wise linearly independent hash function and then use it as a new identity in ABB IBE scheme. Roughly

speaking, d-extended ABB IBE has a master public key size that is d-times longer than the original ABB IBE and has almost the same ciphertext size. We call it d-extended ABB IBE scheme. For d-extended ABB IBE, we can achieve better reduction cost of $O(\epsilon^{1+\frac{1}{d-1}}/qQ)$ using the power of $\mathsf{F}_{\mathsf{ParWat}}$ for arbitrarily chosen odd d.

Theorem 10. *If there is an $(t_{\mathsf{A}}, Q, \epsilon_{\mathsf{A}})$-adversary A against the $\mathsf{IND\text{-}CPA}$ security of the d-extended ABB IBE scheme for odd integer $d \geq 3$, there is an adversary B that breaks the LWE problem with advantage ϵ_{B} and t_{B} such that*

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}^{1+\frac{1}{d-1}}}{12qQ} - \mathsf{negl}(\lambda), \quad t_{\mathsf{B}} = t_{\mathsf{A}} + Q \cdot \mathsf{poly}(\lambda).$$

In particular, if we have $d \geq \omega(1)$, we have

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}}{12q\lambda Q} - \mathsf{negl}(\lambda), \quad t_{\mathsf{B}} = t_{\mathsf{A}} + Q \cdot \mathsf{poly}(\lambda)$$

where $q^n \geq 2 \cdot Q \cdot \epsilon^{-\frac{1}{d-1}}$ holds for dimension n of the scheme and $\mathsf{poly}(\lambda)$ is roughly the overhead incurred by the running the simulated algorithms compared to the real $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt})$ algorithms.

Note that Theorem 9 is a special case of Theorem 10, since d-extended ABB scheme with $d = 3$ equals to the ABB scheme. The proof of Theorem 10 can be obtained by showing that d-extended ABB IBE admits partitioning-based reduction and plugging in our analysis on $\mathsf{F}_{\mathsf{ParWat}}$ in Sec. 5.5 into our template. In App. B, we provide the formal proof of the theorems and necessary background, including the description of ABB and d-extended ABB IBE schemes.

7 Application to VRFs

In this section, we apply the tools we developed in Sec. 4 and 5 to VRF. Similarly to the case of IBE (Sec. 6), we prepare a security proof template that allows us to prove the security of VRF using partitioning function with approximation in a modular manner. However, unlike Sec. 6, we do not focus on applying our framework to existing schemes. Rather, we construct a new VRF scheme and then apply our framework to the scheme. The new VRF scheme subsumes the previous schemes in terms of asymptotic sapce efficiency and security at the same time, in the sense that it is proven secure under the standard d-LIN assumption with tighter reductions. We refer to Table 2 for the overview.

7.1 Security Proof Template for VRF

Similarly to what we have done for IBE schemes, we show a security proof template for VRF schemes using the artificial abort paradigm.

Definition 12 (Partitioning-Based Reduction for VRF)**.** *We say that there is a partioning-based reduction for a VRF scheme $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$ from a decision problem $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ with respect to a partitioning function with approximation $\mathsf{F} = \{\mathsf{F} : \mathcal{K} \times \{0,1\}^{\ell} \to \{0,1\}\}$ if there exists a tuple of efficient simulation algorithms $(\mathsf{SimGen}, \mathsf{SimEval}, \mathsf{SimChal})$ with the following syntax.*

$\mathsf{SimGen}(K, \psi) \to (\mathsf{vk}, \mathsf{td})$. *It takes as input a partitioning key $K \in \mathcal{K}$ and the problem instance ψ (output by either $\mathcal{D}_0$ or $\mathcal{D}_1$) and outputs a verification key vk and a trapdoor td.*

$\mathsf{SimEval}(\mathsf{td}, \mathsf{x}) \to (\mathsf{y}, \pi)$. *It takes as input a trapdoor* td *and an input* $\mathsf{x} \in \{0,1\}^\ell$ *and outputs the value* y *and corresponding proof* π.

$\mathsf{SimChal}(\mathsf{td}, \psi, \mathsf{x}) \to \mathsf{y}$. *It takes as input a trapdoor* td*, a problem instance* ψ *(output by either* $\mathcal{D}_0$ *or* $\mathcal{D}_1$*) and outputs a value* y.

For these algorithms, we require the following properties.

Simulation indistinguishability: *For functions* $Q = Q(\lambda)$ *and* $\epsilon = \epsilon(\lambda)$*, and a PPT adversary* A*, let us define the advantage for the computational verification simulatability as follows:*

$$\mathsf{Adv}^{\mathsf{sim\text{-}ind}}_{\mathsf{PBR}}(\mathsf{A}) = \left| \Pr\left[\mathsf{A}(K, \mathsf{vk})^{\mathsf{Eval}(\mathsf{sk},\cdot)} = 1 : \begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right) \end{array} \right] - \Pr\left[\mathsf{A}(K, \mathsf{vk})^{\mathsf{Sim}(K,\mathsf{td},\psi,\cdot)} = 1 : \begin{array}{c} \psi \leftarrow \mathcal{D}_0 \\ K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right) \\ (\mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{SimGen}(K, \psi) \end{array} \right] \right|,$$

where $\mathsf{Eval}(\mathsf{sk}, \cdot)$ *takes as input* $\mathsf{x} \in \{0,1\}^\ell$ *and returns* $\mathsf{Eval}(\mathsf{sk}, \mathsf{x})$*, and the oracle* $\mathsf{Sim}(K, \mathsf{td}, \psi, \cdot)$ *is defined as follows :*

$\mathsf{Sim}(K, \mathsf{td}, \psi, \cdot)$*: It takes as input* $\mathsf{x} \in \{0,1\}^\ell$ *and returns* $\mathsf{SimEval}(\mathsf{td}, \mathsf{x})$ *if* $\mathsf{F}(K, \mathsf{x}) = 1$ *and* $\mathsf{SimChal}(\mathsf{td}, \psi, \mathsf{x})$ *if* $\mathsf{F}(K, \mathsf{x}) = 0$.

The adversary is allowed to access $\mathsf{Eval}(\mathsf{sk}, \cdot)$ *for* Q *times. We say that the adversary* A *is an* $(t, Q, \epsilon, \epsilon_{\mathsf{A}})$ *adversary if it has an advantage* $\epsilon_{\mathsf{A}} = \mathsf{Adv}^{\mathsf{sim\text{-}ind}}_{\mathsf{PBR}}(\mathsf{A})$ *in the above game. We require that for all polynomial* $Q(\lambda)$*, noticeable* $\epsilon(\lambda)$*, and all PPT adversary* A*, we require* $\epsilon_{\mathsf{A}}(\lambda) = \mathsf{negl}(\lambda)$.

Function value randomizability: *For all* $K \in \mathcal{K}$*, all* $(\mathsf{vk}, \mathsf{td})$ *such that there exists* $\psi \in \mathcal{D}_1$ *satisfying* $(\mathsf{vk}, \mathsf{td}) \in \mathsf{SimGen}(K, \psi)$*, and all* $\mathsf{x} \in \{0,1\}^\ell$ *such that* $\mathsf{F}(K, \mathsf{x}) = 0$*, the following distributions are the same:*

$$\{\mathsf{y} = \mathsf{SimChal}(\mathsf{td}, \psi, \mathsf{x})\} \equiv \left\{\mathsf{y} \xleftarrow{\$} \mathcal{Y}\right\},$$

where ψ *is sampled conditioned on* $(\mathsf{vk}, \mathsf{td}) = \mathsf{SimSetup}(K, \psi)$.

Remark 6 (Definitional differences between IBE and VRF)**.** *The definition of partitioning-based reduction for VRF scheme is more succinct and general compared to that for IBE schemes (see Def. 11). Roughly, the master public key, secret key, and ciphertext simulatability of the IBE scheme is packed into the simulation indistinguishability of the VRF scheme. While this makes the definition more succinct and general, the proof becomes more complex as we need to implicitly prove all three properties in one game. This definitional choice was dictated by the concrete IBE and VRF constructions we handle in this work. Concretely, while we are able to define partitioning-based reduction for VRF scheme more similarly to those of the IBE schemes, we will not be able to prove that for the VRF scheme we construct in Sec. 7.3. This is in particular because the master public key, secret key, and ciphertext simulatability are computationally intertwined and cannot be separated.*

The following establishes the security of a VRF scheme with a partitioning-based reduction. Below, for the construction provided in this work, the dominant runtime overhead of the reduction is T_{approx} as $Q \cdot (T_{\mathsf{F}} + \mathsf{poly}(\lambda)) \lesssim t$.

Theorem 11. *Assume that there is a computational partioning-based reduction for a VRF scheme* $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$ *from a decision problem* $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ *with respect to a* $(\gamma_{\mathsf{min}}, T_{\mathsf{F}}, T_{\mathsf{approx}})$*-partitioning function* F*. Then, if there is an* (t, Q, ϵ)*-adversary* A *against the pseudorandomness of the VRF scheme with a polynomial* Q *and non-negligible* ϵ*, there is an* (ϵ', t')*-adversary* A' *against the problem* $\mathcal{D}$ *and an* (ϵ'', Q, t'')*-adversary* A'' *against the simulation indistinguishability property such that*

$$t', t'' = t + T_{\mathsf{approx}} + Q \cdot (T_{\mathsf{F}} + \mathsf{poly}(\lambda)), \qquad \epsilon' + \epsilon'' \geq \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon,$$

for a non-negligible ϵ', ϵ'' *and infinitely many* $\lambda \in \mathbb{N}$*. Here,* $\mathsf{poly}(\lambda)$ *is roughly the overhead incurred by running the simulated algorithms compared to the real* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$ *algorithms.*

Proof. We prove the theorem by a sequence of games. Let ϵ_i denote the advantage of A in Game_i. Below, we use the fact that since $\epsilon(\lambda)$ is non-negligible, there exists a noticeable function $\epsilon^*(\lambda)$ such that $\epsilon(\lambda) \geq \epsilon^*(\lambda)$ for infinitely many $\lambda \in \mathbb{N}$.

Game_0**:** This is the real pseudorandomness game. By assumption, we have $\epsilon_0 = \epsilon$.

Game_1**:** In this game, we generate the partitioning key $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon^*\right)$ at the end of the game, independently from anything else. Even though we do not embed K into the parameters (i.e., K is information theoretically hidden from the adversary), we introduce the *artificial abort* step here. Note that due to our assumption and Def. 9, Item 1, for large enough λ, we have $K \in \mathcal{K}$ and the properties in Items 2 and 3 hold.

Following an identical analysis given in the proof of Theorem 7, we have

$$\epsilon_1 = \left|\Pr[\mathsf{coin}' = \mathsf{coin}] - \frac{1}{2}\right| \geq \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon \geq \frac{\gamma_{\mathsf{min}}}{3} \cdot \epsilon^*,$$

for infinitely many $\lambda \in \mathbb{N}$.

Moreover, the challenger's runtime overhead compared to Game_0 is $T_{\mathsf{approx}}(Q, \epsilon) + Q \cdot T_{\mathsf{F}}(Q, \epsilon)$.

Game_2**:** In this game, the partitioning key K is chosen at the beginning of the game and once the Bad event in Theorem 1 is satisfied (i.e., $\mathsf{F}(K, \mathsf{ID}^{(i)}) = 0$ for $i \in [Q]$ or $\mathsf{F}(K, \mathsf{ID}^*) = 1$ during the game), the challenger aborts without running the game until the end. This is only a conceptual change and we have $\epsilon_2 = \epsilon_1$.

Game_3**:** In this game, we change the game so that the challenger uses the trapdoor to simulate the game. In more detail, the challenger chooses $\psi \xleftarrow{\$} \mathcal{D}_0$ and runs $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right)$ at the beginning of the game. Then, the challenger answers any evaluation query x by $\mathsf{SimEval}(\mathsf{td}, \mathsf{x})$ and the challenge query x^* by $\mathsf{SimChal}(\mathsf{td}, \psi, \mathsf{x}^*)$ throughout the game.

We can show that the view of the adversary in this game is computationally indistinguishable from the previous game by a straightforward reduction to the simulation indistinguishability of the simulation algorithms $(\mathsf{SimGen}, \mathsf{SimEval}, \mathsf{SimChal})$. Therefore, we can construct an adversary A'' against the simulation indistinguishability property such that $\mathsf{Adv}^{\mathsf{sim\text{-}ind}}_{\mathsf{PBR}}(\mathsf{A}) = |\epsilon_3 - \epsilon_2|$.

Game_4**:** In this game, we sample ψ as $\psi \xleftarrow{\$} \mathcal{D}_1$ instead of $\psi \xleftarrow{\$} \mathcal{D}_0$. The rest of the game is the same as the previous one.

If there is an adversary who can distinguish this game from the previous one, we can construct a distinguisher A against $\mathcal{D}$ such that $\mathsf{Adv}^{\mathcal{D}}(\mathsf{A}) = |\epsilon_4 - \epsilon_3|$.

Game_5**:** In this game, we choose the challenge function value y^* as $\mathsf{y}^* \xleftarrow{\$} \mathcal{Y}$ regardless of the value of coin. By the function value randomizability, this change is conceptual and we have $\epsilon_5 = \epsilon_4$.

In Game_5, coin is information theoretically hidden from A, and hence, $\epsilon_5 = 0$. Collecting all the bounds, we arrive at the theorem statement. □

7.2 Preliminaries

Additional Notations. In this section, we use symmetric pairings and additive notations for them for the sake of simplicity. Concretely, for a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, to describe group elements g^a and $g_T^a = e(g,g)^a$, we denote $[a]$ and $[a]_T$, respectively. We use similar notation for vectors and denote by $[\mathbf{v}]$ the group elements $(g^{v_1}, \ldots, g^{v_d})^\top$ for a vector $\mathbf{v} = (v_1, \ldots, v_d)^\top \in \mathbb{Z}_p^d$. We also use similar notation for matrices. For vectors $\mathbf{v} = (v_1, \ldots, v_d)^\top \in \mathbb{Z}_p^d$ and $\mathbf{w} = (w_1, \ldots, w_d)^\top \in (\mathbb{Z}_p^*)^d$ with the same dimension d, $\mathbf{v} \odot \mathbf{w}$ denotes the vector $(v_1 w_1, \ldots, v_d w_d)^\top$ and $\mathbf{v} \oslash \mathbf{w}$ denotes $(v_1/w_1, \ldots, v_d/w_d)^\top$. It is easy to see that for any matrix $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_d] \in \mathbb{Z}_p^{d \times d}$ and vectors $\mathbf{v} \in \mathbb{Z}_p^d$ and $\mathbf{w} \in \mathbb{Z}_p^d$, it holds that $(\mathbf{B}\mathbf{v}) \oslash \mathbf{w} = [\mathbf{b}_1 \oslash \mathbf{w}, \ldots, \mathbf{b}_d \oslash \mathbf{w}]\mathbf{v}$.[14] Given $[\mathbf{v}]$ and $[\mathbf{w}]$, we can compute $[\mathbf{v} \odot \mathbf{w}]_T$ by the component-wise pairing computation. We denote this by $[\mathbf{v}] \odot [\mathbf{w}]$.

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

Definition 13. *A bilinear group generator is a probabilistic polynomial-time algorithm* GrpGen *that takes as input a security parameter* λ *(in unary) and outputs* $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1)) \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$ *such that the following requirements are satisfied.*

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

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

[14] We note that $(\mathbf{B}\mathbf{v}) \oslash \mathbf{w} \neq \mathbf{B}(\mathbf{v} \oslash \mathbf{w})$ in general. This is the reason why we do not omit the parenthesis from the expression $(\mathbf{B}\mathbf{v}) \oslash \mathbf{w}$.

Definition 14. *We say that group generator* GrpGen *is certified, if there exists a deterministic polynomial-time algorithm* GrpVfy *with the following properties.*

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

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

and all requirements from Def. 13 are satisfied.

Recognition and unique representation of elements of $\mathbb{G}$**.** *Furthermore, we require that each element in* $\mathbb{G}$ *has a unique representation, which can be efficiently recognized. That is, on input two strings* $\mathcal{G}$ *and* s*,* $\mathsf{GrpVfy}(\mathcal{G}, s)$ *outputs* 1 *if and only if* $\mathsf{GrpVfy}(\mathcal{G}) = 1$ *and it holds that* $s = \phi(x)$ *for some* $x \in \mathbb{Z}_p$*. Here* $\phi : \mathbb{Z}_p \to \mathbb{G}$ *denotes the fixed group isomorphism contained in* $\mathcal{G}$ *to specify the representation of elements of* $\mathbb{G}$ *(see Def. 13).*

We recall the definitions of the d-linear (d-LIN) assumption and the d-rank assumption following the presentation by Kohl [Koh19]. We note that d-LIN assumption implies d-rank assumption.

Definition 15 (d-linear Problem)**.** *Let* $\mathcal{G}$ *be a description of bilinear group generated by* GrpGen*. For a PPT algorithm* A*, the advantage of* A *for the* d*-linear problem is defined by*

$$\mathsf{Adv}^{d\text{-lin}}_{\mathcal{G}}(\mathsf{A}) := \left| \Pr\left[\mathsf{A}\left(\mathcal{G}, [\mathbf{c}], [\mathbf{d}], \left[\sum_{i=1}^{d} d_i / c_i \right] \right) = 1 \right] - \Pr[\mathsf{A}(\mathcal{G}, [\mathbf{c}], [\mathbf{d}], [r]) = 1] \right|$$

where $\mathbf{c}, \mathbf{d} \xleftarrow{\$} \mathbb{Z}_p^d$ *and* $r \xleftarrow{\$} \mathbb{Z}_p$*. We say that the* d*-linear (*d*-LIN) assumption holds if* $\mathsf{Adv}^{d\text{-lin}}_{\mathcal{G}}(\mathsf{A})$ *is negligible for all PPT algorithm* A*. We also say that* A *is an* (t, ϵ)*-adversary against the* d*-LIN problem if* A *runs in at most time* t *and satisfies* $\mathsf{Adv}^{d\text{-lin}}_{\mathcal{G}}(\mathsf{A}) \geq \epsilon$*.*

Definition 16 (d-rank Problem)**.** *Let* $\mathcal{G}$ *be a description of bilinear group generated by* GrpGen*. For a PPT algorithm* A*, the advantage of* A *for the* d*-rank problem is defined by*

$$\mathsf{Adv}^{d\text{-rank}}_{\mathcal{G}}(\mathsf{A}) := |\Pr[\mathsf{A}(\mathcal{G}, [\mathbf{M}_{d-1}]) = 1] - \Pr[\mathsf{A}(\mathcal{G}, [\mathbf{M}_d]) = 1]|$$

where $\mathbf{M}_i$ *is uniformly chosen at random from the set of matrices of rank* i *in* $\mathbb{Z}_p^{d \times d}$ *for* $i \in \{d-1, d\}$*. We say that the* d*-rank assumption holds if* $\mathsf{Adv}^{d\text{-rank}}_{\mathcal{G}}(\mathsf{A})$ *is negligible for all PPT algorithm* A*. We also say that* A *is an* (t, ϵ)*-adversary against the* d*-rank problem if* A *runs in at most time* t *and satisfies* $\mathsf{Adv}^{d\text{-rank}}_{\mathcal{G}}(\mathsf{A}) \geq \epsilon$*.*

7.3 Our New Short VRF

Here, we propose new construction of VRF with short parameters. Our scheme achieves the best space efficiency among the existing schemes and enjoys the security proof under a static assumption at the same time. Our construction is based on the the construction proposed by Kohl [Koh19], but we substantially improve the space efficiency by adding a new twist to the scheme. We then proceed to prove the security of the scheme based on our framework. Our framework yields tighter reduction cost compared to the conventional analyses.

In Fig. 1, we give the description of our new VRF scheme. For the construction, we need an error correcting code $\mathsf{Encode} : \{0,1\}^\ell \to \Sigma^n$ for an alphabet Σ and an injective map $\mathsf{Inj} : [n] \times \Sigma \to [n_1] \times [n_2]$. In order to be able to define such an injective map, we need to have $n|\Sigma| \leq n_1 n_2$, where $|\Sigma|$ is the size of the alphabet. We will typically set $n_1 = n_2 = \lceil \sqrt{n|\Sigma|} \rceil$ to achieve the

smallest verification key size. For the construction, we use the map $\mathsf{S} : \{0,1\}^{\ell} \to 2^{[n_1]\times[n_2]}$ defined as

$$\mathsf{S}(\mathsf{x}) := \{\ \mathsf{Inj}(i, \mathsf{Encode}(\mathsf{x})_i) : i \in [n]\ \},$$

where $\mathsf{Encode}(\mathsf{x})_i \in \Sigma$ denotes the i-th symbol of $\mathsf{Encode}(\mathsf{x}) \in \Sigma^n$. We can instantiate Encode by the binary or non-binary error correcting codes provided in Lemma 5. As we will discuss in Sec. 7.5, different choice of error correcting codes leads to trade-offs between the efficiency and the reduction loss. The construction is parameterized by d and is secure under the d-LIN assumption similarly to [Koh19]. We typically choose d to be small constant like $d = 2$ or $d = 3$.

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

1 : $\mathcal{G} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$

2 : $\mathbf{M}_{i,j} \xleftarrow{\$} \mathbb{Z}_p^{d\times d}$ for $i \in [\eta]$ and $j \in [n_1]$

3 : $\mathbf{N}_{i,k} \xleftarrow{\$} \mathbb{Z}_p^{d\times d}$ for $i \in [\eta]$ and $k \in [n_2]$

4 : $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^d \backslash \{\mathbf{0}_d\}, \mathbf{w} \xleftarrow{\$} (\mathbb{Z}_p^*)^d$

5 : $\mathsf{vk} := (\mathcal{G}, [\mathbf{u}], [\mathbf{w}], \{[\mathbf{M}_{i,j}]\}_{\substack{i\in[\eta]\\ j\in[n_1]}}, \{[\mathbf{N}_{i,k}]\}_{\substack{i\in[\eta]\\ k\in[n_2]}})$

6 : $\mathsf{sk} := (\mathcal{G}, \mathbf{u}, \mathbf{w}, \{\mathbf{M}_{i,j}\}_{\substack{i\in[\eta]\\ j\in[n_1]}}, \{\mathbf{N}_{i,k}\}_{\substack{i\in[\eta]\\ k\in[n_2]}})$

7 : **return** $(\mathsf{vk}, \mathsf{sk})$

$\mathsf{Eval}(\mathsf{sk}, \mathsf{x})$

1 : **parse** $\mathsf{sk} \leftarrow (\mathcal{G}, \mathbf{u}, \mathbf{w}, \{\mathbf{M}_{i,j}\}_{i,j}, \{\mathbf{N}_{i,k}\}_{i,k})$

2 : Compute $\mathsf{S}(\mathsf{x}) \subseteq [n_1] \times [n_2]$

3 : Compute $\mathbf{P}_i := \sum_{(j,k)\in\mathsf{S}(\mathsf{x})} \mathbf{M}_{i,j}\mathbf{N}_{i,k}$ for $i \in [\eta]$

4 : Compute $\mathbf{v}_i := \left(\prod_{\iota=1}^{i} \mathbf{P}_\iota\right)^\top \mathbf{u}$ for $i \in [\eta]$

5 : $\mathbf{z} := \mathbf{v}_\eta \oslash \mathbf{w}$

6 : $\mathsf{y} := [\langle \mathbf{z}, \mathbf{1}_d \rangle]$, $\pi := (\{[\mathbf{v}_i], [\mathbf{P}_i]\}_{i\in[\eta]}, [\mathbf{z}])$

7 : **return** (y, π)

$\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \mathsf{y}, \pi)$

1 : Check that vk is in the following form and output 0 otherwise:
$\mathsf{vk} = (\mathcal{G}, [\mathbf{u}] \in \mathbb{G}^d, [\mathbf{w}] \in \mathbb{G}^d, \{[\mathbf{M}_{i,j}] \in \mathbb{G}^{d\times d}\}_{i\in[\eta],j\in[n_1]}, \{[\mathbf{N}_{i,k}] \in \mathbb{G}^{d\times d}\}_{i\in[\eta],k\in[n_2]})$
such that $\mathsf{GrpVfy}(\mathcal{G}) = 1$

2 : Check that y and π are in the following form and output 0 otherwise:
$\mathsf{y} \in \mathbb{G}, \quad \pi = (\{[\mathbf{v}_i] \in \mathbb{G}^d, [\mathbf{P}_i] \in \mathbb{G}^{d\times d}\}_{i\in[\eta]}, [\mathbf{z}] \in \mathbb{G}^d)$

3 : Check whether the following equations hold for all $i \in [\eta]$ and output 0 otherwise:
$e([\mathbf{I}_d], [\mathbf{P}_i]) \stackrel{?}{=} \prod_{(j,k)\in\mathsf{S}(\mathsf{x})} e([\mathbf{M}_{i,j}], [\mathbf{N}_{i,k}])$, $e([\mathbf{I}_d], [\mathbf{v}_i]) \stackrel{?}{=} e([\mathbf{P}_i^\top], [\mathbf{v}_{i-1}])$, where $\mathbf{v}_0 := \mathbf{u}$

4 : Check whether the following equations hold and output 0 otherwise:
$[\mathbf{z}] \odot [\mathbf{w}] \stackrel{?}{=} [\mathbf{1}_d] \odot [\mathbf{v}_\eta]$, $\mathsf{y} \stackrel{?}{=} [\langle \mathbf{z}, \mathbf{1}_d \rangle]$

5 : **return** 1

Figure 1: Our VRF Scheme.

7.4 Correctness, Unique Provability, and Pseudorandomness

Here, we prove correctness, unique provability, and pseudorandomness of our scheme.

Correctness. We prove correctness of the scheme. It is easy to see that an honestly generated proof passes all the verification steps except for the check $e([\mathbf{I}_d], [\mathbf{v}_i]) \stackrel{?}{=} e([\mathbf{P}_i^\top], [\mathbf{v}_{i-1}])$ for $i \in [\eta]$.

We show that the proof also passes the check as well. This can be seen by observing

$$\mathbf{v}_i = \left(\prod_{\iota=1}^{i} \mathbf{P}_\iota\right)^\top \mathbf{u} = \mathbf{P}_i^\top \left(\prod_{\iota=1}^{i-1} \mathbf{P}_\iota\right)^\top \mathbf{u} = \mathbf{P}_i^\top \mathbf{v}_{i-1}$$

holds for all $i \in [\eta]$.

Unique Provability. We prove the unique provability of the scheme. We first observe that for each $i \in [\eta]$, there is unique $\mathbf{P}_i \in \mathbb{Z}_p^{d\times d}$ that satisfies $[\mathbf{P}_i] = \prod_{(j,k)\in \mathsf{S}(\mathsf{x})} e([\mathbf{M}_{i,j}], [\mathbf{N}_{i,k}])$. Therefore, there is unique sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\eta \in \mathbb{Z}_p^d$ that satisfies $e([\mathbf{I}_d], [\mathbf{v}_i]) = e([\mathbf{P}_i^\top], [\mathbf{v}_{i-1}])$ for all $i \in [\eta]$. This in particular implies that $\mathbf{v}_\eta$ that passes the verification is unique and thus $\mathbf{z}$ that satisfies $[\mathbf{z}] \odot [\mathbf{w}] = [\mathbf{1}_d] \odot [\mathbf{v}_\eta]$ is unique. Since $\mathbf{z}$ is unique, $\langle \mathbf{z}, \mathbf{1}_d \rangle$ is unique as well. Finally, as the group described by $\mathcal{G}$ satisfies recognition and unique representation of group elements, unique provability follows.

Pseudorandomness. The following theorem addresses the pseudorandomness of the scheme.

Theorem 12. *If there is an $(t_{\mathsf{A}}, Q, \epsilon_{\mathsf{A}})$-adversary A against the pseudorandomness of the our VRF scheme in Fig. 1 instantiated with $\mathsf{Encode} : \{0,1\}^\ell \to \Sigma^n$ that has small triple overlap property as per Def. 10 with parameter c, there is an $(\epsilon_{\mathsf{B}}, t_{\mathsf{B}})$-adversary B against the d-LIN problem and an $(\epsilon_{\mathsf{B}'}, t_{\mathsf{B}'})$-adversary B' against the d-rank problem such that*

$$t_{\mathsf{B}}, t_{\mathsf{B}'} = t_{\mathsf{A}} + Q \cdot \mathsf{poly}(\lambda, n), \quad \epsilon_{\mathsf{B}} + 2\eta\epsilon_{\mathsf{B}'} \geq \frac{\gamma_{\mathsf{min}}\epsilon_{\mathsf{A}}}{6|\Sigma|^{\eta'}} - \frac{\eta d(n_1+n_2)}{p},$$

where $\eta = \omega(\log(\lambda))/\log(1/1-c)$, $\eta' = \lceil \log(2Q/\sqrt{\epsilon})/\log(1/1-c) \rceil$, and $\mathsf{poly}(\lambda, n)$ is a fixed polynomial independent from Q and ϵ_{A}.

To prove the theorem, we use our template introduced in Sec. 7.1. Namely, we define the algorithms $(\mathsf{SimGen}, \mathsf{SimEval}, \mathsf{SimChal})$ that are associated with the partitioning function with approximation $\mathsf{F}_{\mathsf{SSM}}$ defined in Sec. 5.6 in the following. Rest of this section is devoted to prove function randomizability of the simulation algorithms (Theorem 13) and simulation indistinguishability (Theorem 14). The above theorem follows immediately from these properties.

$\mathsf{SimGen}(K, \psi)$: It takes as input the partitioning key K and a problem instance ψ. It then parses them as $K \to \{(I_i, \sigma_i)\}_{i\in[\eta']}$ and $\psi \to (\mathcal{G}, [\mathbf{c}], [\mathbf{d}], [t])$, where we have $I_i \in [n]$ and $\sigma_i \in \Sigma$ for all $i \in [\eta']$ and $\eta' \leq \eta$. It first discards $[t]$ and then works as follows.

- It samples $\mathbf{a}_1, \ldots, \mathbf{a}_{d-1} \xleftarrow{\$} \mathbb{Z}_p^d$ and sets $[\mathbf{B}] := [\mathbf{a}_1 \odot \mathbf{c}, \cdots, \mathbf{a}_{d-1} \odot \mathbf{c}, \mathbf{d}] \in \mathbb{Z}_p^{d\times d}$.
- For each $i \in \{0, 1, \ldots, \eta-1\}$, it chooses subspaces $\mathcal{U}_i$ and $\mathcal{V}_i$ of dimension $d-1$ independently and uniformly at random. Furthermore, $\mathcal{U}_\eta$ is defined to be the subspace spanned by the first $d-1$ unit vectors.
- It chooses $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^d \backslash \mathcal{U}_0$ and sets $\mathbf{w} := \mathbf{c}$.
- It computes $(j_i, k_i) := \mathsf{Inj}(I_i, \sigma_i)$ for all $i \in [\eta']$ and sets $(j_i, k_i) := (j_1, k_1)$ for $i \in [\eta'+1, \eta]$.

It then chooses $\{\mathbf{M}_{i,j} \in \mathbb{Z}_p^{d\times d}\}_{i\in[\eta], j\in[n_1]}$ and $\{\mathbf{R}_{i,k} \in \mathbb{Z}_p^{d\times d}\}_{i\in[\eta], k\in[n_2]}$ as follows.

- For $i \in [\eta]$, the algorithm samples $\mathbf{M}_{i,j_i}$ and $\mathbf{R}_{i,k_i}$ uniformly of rank d subject to

$$\mathcal{U}_i = \mathbf{R}_{i,k_i}^\top \mathcal{V}_{i-1}, \qquad \mathcal{V}_{i-1} = \mathbf{M}_{i,j_i}^\top \mathcal{U}_{i-1}.$$

– For other i, j, and k, the algorithm samples $\mathbf{M}_{i,j}$ and $\mathbf{R}_{i,k}$ uniformly of rank $d-1$ subject to

$$\mathcal{U}_i = \mathbf{R}_{i,k}^\top \mathcal{V}_{i-1}, \qquad \mathcal{V}_{i-1} = \mathbf{M}_{i,j}^\top \mathcal{U}_{i-1}.$$

Finally, the algorithm sets

$$[\mathbf{N}_{i,k}] := \begin{cases} [\mathbf{R}_{i,k}] & \text{if } i \in [\eta - 1] \\ [\mathbf{R}_{i,k}] \cdot [\mathbf{B}]^\top & \text{if } i = \eta \end{cases}$$

for all $k \in [n_2]$. Finally, it outputs the verification key

$$\mathsf{vk} := (\mathcal{G}, [\mathbf{u}], [\mathbf{w}], \{[\mathbf{M}_{i,j}]\}_{i\in[\eta],j\in[n_1]}, \{[\mathbf{N}_{i,k}]\}_{i\in[\eta],k\in[n_2]})$$

and the trapdoor

$$\mathsf{td} := (\mathcal{G}, \mathbf{u}, \mathbf{a}_1, \ldots, \mathbf{a}_{d-1}, [\mathbf{B}], [\mathbf{c}], [\mathbf{d}], \{\mathbf{M}_{i,j}\}_{i\in[\eta],j\in[n_1]}, \{\mathbf{R}_{i,k}\}_{i\in[\eta],k\in[n_2]}). \tag{46}$$

Note that SimGen discards the challenge term $[t]$ of the problem instance ψ.

$\mathsf{SimEval}(\mathsf{td}, \mathsf{x}) \to (\mathsf{y}, \pi)$. It parses td as Eq. (46) and runs as follows.

– For $i \in [\eta]$, it computes $\mathbf{Q}_i$ as $\mathbf{Q}_i := \sum_{(j,k)\in\mathsf{S}(\mathsf{x})} \mathbf{M}_{i,j}\mathbf{R}_{i,k}$.

– It computes

$$\mathbf{b} = (b_1, \ldots, b_d)^\top := \left(\prod_{\iota=1}^{\eta} \mathbf{Q}_\iota\right)^\top \mathbf{u} \in \mathbb{Z}_p^d. \tag{47}$$

– For $i \in [\eta]$, it computes $\mathbf{P}_i$ and $\mathbf{v}_i$ as

$$[\mathbf{P}_i] = \begin{cases} [\mathbf{Q}_i] & \text{if } i \le \eta - 1 \\ [\mathbf{Q}_\eta] \cdot [\mathbf{B}]^\top & \text{if } i = \eta \end{cases}, \qquad [\mathbf{v}_i] = \begin{cases} \left[\left(\prod_{\iota=1}^{i} \mathbf{Q}_\iota\right)\right]^\top \mathbf{u} & \text{if } i \le \eta - 1 \\ [\mathbf{B}] \cdot \mathbf{b} & \text{if } i = \eta \end{cases}. \tag{48}$$

– It computes $\mathbf{z}$ as $\mathbf{z} := \sum_{i=1}^{d-1} b_i \mathbf{a}_i$.

– Finally, it outputs $\mathsf{y} := [\langle \mathbf{z}, \mathbf{1}_d \rangle]$ and $\pi := (\{[\mathbf{v}_i], [\mathbf{P}_i]\}_{i\in[\eta]}, [\mathbf{z}])$.

$\mathsf{SimChal}(\mathsf{td}, \psi, \mathsf{x}) \to \mathsf{y}$. It parses ψ as $\psi \to (\mathcal{G}, [\mathbf{c}], [\mathbf{d}], [t])$ and td as Eq. (46). It then computes $\mathbf{b}$ as in Eq. (47). It then computes y as

$$\mathsf{y} := \left[b_d t + \left\langle \mathbf{1}_d, \sum_{i=1}^{d-1} b_i \mathbf{a}_i \right\rangle \right] \tag{49}$$

using $[t]$. Finally, it outputs y.

Before proving function value randomizability and the simulation indistinguishability of the above algorithms, we prove the following useful lemma.

Lemma 6. *For all $K \in \mathcal{K}_\lambda$, $\psi \in \mathcal{D}_0 \cup \mathcal{D}_1$, $(\mathsf{vk}, \mathsf{td}) \in \mathsf{SimGen}(K, \psi)$, $\mathsf{x} \in \{0,1\}^\ell$, and $\mathbf{b}$ computed as in Eq. (47), we have $b_d = 0$ if and only if $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$.*

Proof. We first prove that the following holds for all $i \in [\eta]$:

$$\mathbf{Q}_i^\top \mathcal{U}_{i-1} \subseteq \mathcal{U}_i. \tag{50}$$

This can be seen by observing that $\mathbf{Q}_i^\top \mathcal{U}_{i-1} = \sum_{(j,k)\in \mathsf{S}(\mathsf{x})} \mathbf{R}_{i,k}^\top \mathbf{M}_{i,j}^\top \mathcal{U}_{i-1}$ and $\mathbf{R}_{i,k}^\top \mathbf{M}_{i,j}^\top \mathcal{U}_{i-1} = \mathcal{U}_i$ for each j and k, where the latter follows from $\mathbf{M}_{i,j}^\top \mathcal{U}_{i-1} = \mathcal{V}_{i-1}$ and $\mathbf{R}_{i,k}^\top \mathcal{V}_{i-1} = \mathcal{U}_i$. We also prove the following equation for all $i \in [\eta]$:

$$\mathbf{Q}_i^\top \left(\mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}\right) \subseteq \begin{cases} \mathcal{U}_i & \text{if } (j_i, k_i) \notin \mathsf{S}(\mathsf{x}) \\ \mathbb{Z}_p^d \backslash \mathcal{U}_i & \text{if } (j_i, k_i) \in \mathsf{S}(\mathsf{x}). \end{cases} \tag{51}$$

Since $\mathbf{Q}_i^\top \left(\mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}\right) = \sum_{(j,k)\in \mathsf{S}(\mathsf{x})} \mathbf{R}_{i,k}^\top \mathbf{M}_{i,j}^\top \left(\mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}\right)$, to prove Eq. (51), it suffices to show that the following equation holds for all $i \in [\eta]$:

$$\mathbf{R}_{i,k}^\top \mathbf{M}_{i,j}^\top \left(\mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}\right) \subseteq \begin{cases} \mathcal{U}_i & \text{if } (j,k) \neq (j_i, k_i) \\ \mathbb{Z}_p^d \backslash \mathcal{U}_i & \text{if } (j,k) = (j_i, k_i) \end{cases}. \tag{52}$$

We prove Eq. (52) by the following case analysis.

The case of $j \neq j_i$. In this case, we have $\mathbf{M}_{i,j}^\top (\mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}) \subseteq \mathbf{M}_{i,j}^\top \mathbb{Z}_p^d = \mathcal{V}_{i-1}$. Since we have $\mathbf{R}_{i,k}^\top \mathcal{V}_{i-1} = \mathcal{U}_i$, the claim follows.

The case of $k \neq k_i$. In this case, the claim follows from $\mathbf{R}_{i,k}^\top \mathbb{Z}_p^d = \mathcal{U}_i$ directly.

The case of $(j,k) = (j_i, k_i)$. We first show that $\mathbf{M}_{i,j_i}^\top \left(\mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}\right) \subseteq \mathbb{Z}_p^d \backslash \mathcal{V}_{i-1}$. For the sake of contradiction, suppose this does not hold. Then there exists a vector $\mathbf{a} \in \mathbb{Z}_p^d \backslash \mathcal{U}_{i-1}$ such that $\mathbf{M}_{i,j_i}^\top \mathbf{a} \in \mathcal{V}_{i-1}$. However, this contradicts the fact that $\mathbf{M}_{i,j_i}$ is full-rank, since $\mathbf{a}$ together with $\mathcal{U}_{i-1}$ spans the entire space $\mathbb{Z}_p^d$ and thus implies $\mathbf{M}_{i,j_i} \mathbb{Z}_p^d \subseteq \mathcal{V}_{i-1}$. Because of the same reasoning, $\mathbf{R}_{i,k_i}^\top \left(\mathbb{Z}_p^d \backslash \mathcal{V}_{i-1}\right) \subseteq \mathbb{Z}_p^d \backslash \mathcal{U}_i$ holds. The claim thus follows.

We also observe that the following holds:

$$\begin{aligned} \mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 0 \quad &\Leftrightarrow \quad \sigma_i = \mathsf{Encode}(\mathsf{x})_{I_i} \quad \forall i \in [\eta'] \\ &\Leftrightarrow \quad \mathsf{Inj}(I_i, \sigma_i) = \mathsf{Inj}(I_i, \mathsf{Encode}(\mathsf{x})_{I_i}) \quad \forall i \in [\eta'] \\ &\Leftrightarrow \quad (j_i, k_i) \in \mathsf{S}(\mathsf{x}) \quad \forall i \in [\eta'] \\ &\Leftrightarrow \quad (j_i, k_i) \in \mathsf{S}(\mathsf{x}) \quad \forall i \in [\eta] \end{aligned} \tag{53}$$

where the first line follows from the definition of $\mathsf{F}_{\mathsf{SSM}}$, the second line follows trivially, (in particular, the only if direction of) the third line follows from the injectivity of Inj, and the last line follows from the definition that $(j_i, k_i) = (j_1, k_1)$ for $i > \eta'$.

We then prove that $\mathbf{b} \in \mathcal{U}_\eta$ if and only if $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$. Recalling that $\mathcal{U}_\eta$ is the space spanned by the first $d-1$ unit vectors, this completes the proof of the lemma. There are two cases to consider.

The case of $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 0$. By Eq. (53), we have $(j_i, k_i) \in \mathsf{S}(\mathsf{x})$ for all $i \in [\eta]$. Then, straightforward induction shows that we have $\mathbf{v}_i \in \mathbb{Z}_p^d \backslash \mathcal{U}_i$ for $i \in [0, \eta - 1]$ and $\mathbf{b} \in \mathbb{Z}_p^d \backslash \mathcal{U}_\eta$, where the base case holds for $i = 0$, and the induction step follows from Eq. (51), with the final step of the induction applied to $\mathbf{v}_{\eta-1}$ and $\mathbf{b} = \mathbf{Q}_\eta \mathbf{v}_{\eta-1}$.

The case of $\mathsf{F}_{\mathsf{SSM}}(K,\mathsf{x})=1$. By Eq. (53), there exists $i^* \in [\eta]$ such that $(j_{i^*},k_{i^*}) \notin \mathsf{S}(\mathsf{x})$. If $i^*=\eta$, $\mathbf{b} \in \mathcal{U}_\eta$ follows from Eq. (50) and (51) directly, which imply $\mathbf{Q}_\eta \mathbb{Z}_p^d \subseteq \mathcal{U}_\eta$. If $i^* \leq \eta-1$, $\mathbf{v}_{i^*} \in \mathcal{U}_{i^*}$ follows from Eq. (50) and (51), which imply $\mathbf{Q}_{i^*}\mathbb{Z}_p^d \subseteq \mathcal{U}_{i^*}$. Then, straightforward induction shows that we have $\mathbf{v}_i \in \mathcal{U}_i$ for $i \in [i^*,\eta-1]$ and $\mathbf{b} \in \mathcal{U}_\eta$, where the base case holds for $i=i^*$, and the induction step follows from Eq. (50), with the final step of the induction applied to $\mathbf{v}_{\eta-1}$ and $\mathbf{b}=\mathbf{Q}_\eta \mathbf{v}_{\eta-1}$.

This concludes the proof of Lemma 6. □

We then prove the function value randomizability of the above algorithms.

Theorem 13. *The simulation algorithms* $(\mathsf{SimGen},\mathsf{SimEval},\mathsf{SimChal})$ *satisfy the function value randomizability as per Def. 12.*

Proof. Since SimGen discards $[t]$, $[t]$ is distributed uniformly at random over $\mathbb{G}$ independently from $(\mathsf{vk},\mathsf{td})$ when $\psi \in \mathcal{D}_1$. Furthermore, it follows by Lemma 6 that $b_d \neq 0$ holds for $\mathbf{b}$ computed as Eq. (47). Therefore, y computed as Eq. (49) is distributed uniformly at random over $\mathbb{G}$, since $[b_d t]$ effectively functions as a one-time pad. □

We then prove the simulation indistinguishability of the above algorithms.

Theorem 14. *If there is* $(t_{\mathsf{A}},Q,\epsilon,\epsilon_{\mathsf{A}},)$*-adversary* A *against the simulation indistinguishability property, there is an* $(\epsilon_{\mathsf{B}'},t_{\mathsf{B}'})$*-adversary* B' *against the* d*-rank problem such that*

$$t_{\mathsf{B}'}=t_{\mathsf{A}}+Q\cdot \mathsf{poly}(\lambda,n), \quad 2\eta\epsilon_{\mathsf{B}'} \geq \epsilon_{\mathsf{A}}-\frac{\eta d(n_1+n_2)}{p},$$

where $\mathsf{poly}(\lambda,n)$ *is a fixed polynomial independent from* Q *and* ϵ_{A}*.*

Proof. Let us fix a PPT adversary A. We prove the theorem using a sequence of games. In the following, let E_{xx} be the probability that the adversary A outputs 1 at the end of $\mathsf{Game}_{\mathrm{xx}}$.

Game_0**:** This is the game where the challenger runs $(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and simulates the oracle $\mathsf{Eval}(\mathsf{sk},\cdot)$ for A.

Game_1**:** In this game, the challenger chooses $\psi \xleftarrow{\$} \mathcal{D}_0$ and $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda,Q,\epsilon\right)$. Then, it chooses subspaces $\mathcal{U}_i$ and $\mathcal{V}_i$ for each $i \in \{0,1,\ldots,\eta-1\}$ and $\mathbf{B}$ as in $\mathsf{SimGen}(K,\psi)$. However, they are ignored throughout the game. Clearly, we have $\Pr[\mathsf{E}_1]=\Pr[\mathsf{E}_0]$.

Game_2**:** In this game, we sample $\mathbf{u}$ as $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^d \backslash \mathcal{U}_0$ instead of $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^d \backslash \{\mathbf{0}_d\}$. Since $\mathcal{U}_0$ is never used in the game except for the sampling of $\mathbf{u}$ and thus is information theoretically hidden, this change does not alter the view of the adversary. Therefore, we have $\Pr[\mathsf{E}_2]=\Pr[\mathsf{E}_1]$.

We consider $\mathsf{Game}_{3,\kappa,1}$ for $0 \leq \kappa \leq \eta$ and $\mathsf{Game}_{3,\kappa,2}$ for $0 \leq \kappa \leq \eta-1$ defined as follows:

$\mathsf{Game}_{3,\kappa,1}$ **:** In this game, $\mathbf{M}_{i,j}$ and $\mathbf{N}_{i,k}$ for all $i \leq \kappa$ are chosen as in SimGen. For $i \geq \kappa+1$, they are chosen from $\mathbb{Z}_p^{d\times d}$ uniformly at random as in $\mathsf{Gen}(1^\lambda)$. Note that we only change the distribution of $(\{\mathbf{M}_{i,j}\}_{i,j},\{\mathbf{N}_{i,k}\}_{i,k})$ here and the oracle given to the adversary is still $\mathsf{Eval}(\mathsf{sk},\cdot)$.

$\mathsf{Game}_{3,\kappa,2}$**:** This game is the same as $\mathsf{Game}_{3,\kappa,1}$ except that $\mathbf{M}_{\kappa+1,j}$ are chosen as in SimGen.

Clearly, we have $\Pr[\mathsf{E}_{3,0,1}] = \Pr[\mathsf{E}_2]$, since $\mathsf{Game}_{3,0,1}$ and Game_2 are equivalent. We will show that $|\Pr[\mathsf{E}_{3,\kappa,1}] - \Pr[\mathsf{E}_{3,\kappa,2}]|$ and $|\Pr[\mathsf{E}_{3,\kappa,2}] - \Pr[\mathsf{E}_{3,\kappa+1,1}]|$ are negligible for all κ assuming the hardness of the d-rank problem in the proof of Lemma 7 and 8, respectively.

Game_4**:** This is the game where the challenger runs $(\mathsf{vk}, \mathsf{td}) \xleftarrow{\$} \mathsf{SimGen}(K, \psi)$ and answers the queries made by A by simulating the oracle $\mathsf{Sim}(K, \mathsf{td}, \psi, \cdot)$. In Lemma 9, we will show that $\mathsf{Game}_{3,\eta,1}$ is equivalent to Game_4 and thus we have $\Pr[\mathsf{E}_{3,\eta,1}] = \Pr[\mathsf{E}_4]$.

We can see that the advantage of A against the simulation indistinguishability is $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_4]|$. By the triangle inequality and Lemma 9, we have

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_4]| \leq \sum_{\kappa=0}^{\eta-1} |\Pr[\mathsf{E}_{3,\kappa,1}] - \Pr[\mathsf{E}_{3,\kappa,2}]| + \sum_{\kappa=0}^{\eta-1} |\Pr[\mathsf{E}_{3,\kappa,2}] - \Pr[\mathsf{E}_{3,\kappa+1,1}]|.$$

Therefore, to complete the proof of Theorem 14, it suffice to prove Lemma 7, 8, and 9. The proofs of these lemmas closely follow those of [HJ16, Koh19].

Lemma 7. *For all $\kappa \in \{0, 1, \ldots, \eta-1\}$, there exists an adversary B whose advantage against the d-rank problem is at least $|\Pr[\mathsf{E}_{3,\kappa,1}] - \Pr[\mathsf{E}_{3,\kappa,2}]| - dn_1/p$.*

Proof. We describe an adversary B that uses A to break the d-rank problem with advantage at least $|\Pr[\mathsf{E}_{3,\kappa,1}] - \Pr[\mathsf{E}_{3,\kappa,2}]| - dn_1/p$. B works as follows.

B is given the problem instance $(\mathcal{G}, [\mathbf{A}])$ of the d-rank problem and simulates vk as follows.

1. B first samples $\mathcal{U}_0, \mathcal{U}_1, \ldots, \mathcal{U}_\kappa$ and $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_{\kappa-1}$ as in $\mathsf{Game}_{3,\kappa,1}$. It also samples $K \xleftarrow{\$} \mathsf{PrtSmp}\left(1^\lambda, Q, \epsilon\right)$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^d \backslash \mathcal{U}_0$, and $\mathbf{w} \xleftarrow{\$} (\mathbb{Z}_p^*)^d$.

2. It then samples $\{\mathbf{M}_{i,j}\}_{i\in[\kappa],j\in[n_1]}$ and $\{\mathbf{R}_{i,k}\}_{i\in[\kappa],k\in[n_2]}$ as in $\mathsf{Game}_{3,\kappa,1}$. Note that this can be done efficiently since it sampled $\mathcal{U}_1, \ldots, \mathcal{U}_\kappa$ and $\mathcal{V}_1, \ldots, \mathcal{V}_{\kappa-1}$ by itself. It then sets $\mathbf{N}_{i,k} = \mathbf{R}_{i,k}$ for all $i \in [\kappa]$ and $k \in [n_2]$. Furthermore, it samples $\{\mathbf{M}_{i,j}\}_{i\in[\kappa+2,j],j\in[n_1]}$ (when $\kappa \leq \eta-2$) and $\{\mathbf{R}_{i,k}\}_{i\in[\kappa+1],k\in[n_2]}$ uniformly at random over $\mathbb{Z}_p^{d\times d}$ as in $\mathsf{Game}_{3,\kappa,1}$. We describe how to sample the remaining terms $\{[\mathbf{M}_{\kappa+1,j}]\}_{j\in[n_1]}$ in the next item.

3. B chooses (not necessarily random) $d-1$ linearly independent vectors $\mathbf{e}_1, \ldots, \mathbf{e}_{d-1} \in \mathcal{U}_\kappa$ and a vector $\mathbf{e}_d \in \mathbb{Z}_p^d \backslash \mathcal{U}_\kappa$ and forms an invertible matrix $\mathbf{E} := (\mathbf{e}_1 | \cdots | \mathbf{e}_d)$. It also samples $\mathbf{g}_j^1, \ldots, \mathbf{g}_j^d \xleftarrow{\$} \mathbb{Z}_p^d$ for $j \in [n_1]$. Then, it computes $\{[\mathbf{F}_j]\}_{j\in[n_1]}$ as follows.

   - For $j \in [n_1] \backslash \{j_{\kappa+1}\}$, it implicitly sets $\mathbf{F}_j := (\mathbf{f}_j^1 | \cdots | \mathbf{f}_j^d)$, where $\mathbf{f}_j^1 := \mathbf{A}\mathbf{g}_j^1, \ldots, \mathbf{f}_j^d := \mathbf{A}\mathbf{g}_j^d$. B can compute $[\mathbf{F}_j]$ since it knows $[\mathbf{A}]$ and $\mathbf{g}_j^1, \ldots, \mathbf{g}_j^d$.
   - For $j = j_{\kappa+1}$. it samples $\mathbf{f}_{j_{\kappa+1}}^d \xleftarrow{\$} \mathbb{Z}_p^d$. It then implicitly sets $\mathbf{F}_{j_{\kappa+1}} := (\mathbf{f}_{j_{\kappa+1}}^1 | \cdots | \mathbf{f}_{j_{\kappa+1}}^d)$, where $\mathbf{f}_{j_{\kappa+1}}^1 := \mathbf{A}\mathbf{g}_{j_{\kappa+1}}^1, \ldots, \mathbf{f}_{j_{\kappa+1}}^{d-1} := \mathbf{A}\mathbf{g}_{j_{\kappa+1}}^{d-1}$. Similarly to the above case, B can compute $\left[\mathbf{F}_{j_{\kappa+1}}\right]$ from $[\mathbf{A}]$ and $\mathbf{g}_{j_{\kappa+1}}^1, \ldots, \mathbf{g}_{j_{\kappa+1}}^{d-1}$.

   It then computes $[\mathbf{M}_{\kappa+1,j}]$ for $j \in [n_2]$ as $\mathbf{M}_{\kappa+1,j} := (\mathbf{F}_j \mathbf{E}^{-1})^\top$.

4. Finally, B sets $\mathsf{vk} := (\mathcal{G}, [\mathbf{u}], [\mathbf{w}], \{[\mathbf{M}_{i,j}]\}_{i,j}, \{[\mathbf{N}_{i,k}]\}_{i,k})$ and gives it to A.

When A makes an oracle query x, B answers the query as follows.

1. For $i \in \{1, 2, \ldots, \eta\} \backslash \{\kappa+1\}$, it computes $\mathbf{P}_i = \sum_{(j,k)\in \mathsf{S}(\mathsf{x})} \mathbf{M}_{i,j}\mathbf{N}_{i,k}$. This is possible since $\mathbf{B}$ knows $\mathbf{M}_{i,j}$ and $\mathbf{N}_{i,k}$ in the clear (i.e., not on the exponent) for all j and k when $i \neq \kappa + 1$.
2. It then computes $[\mathbf{P}_{\kappa+1}] = \left[\sum_{(j,k)\in \mathsf{S}(\mathsf{x})} \mathbf{M}_{\kappa+1,j}\mathbf{N}_{\kappa+1,k}\right]$. This can be computed efficiently, since B knows $[\mathbf{M}_{\kappa+1,j}]$ for all j and $\mathbf{N}_{\kappa+1,k}$ for all k in the clear.
3. It computes $[\mathbf{v}_i] = \left[\left(\prod_{\iota=1}^{i} \mathbf{P}_\iota\right)^\top \mathbf{u}\right]$ for $i \in [\eta]$. This can be computed efficiently, since it knows $\mathbf{P}_i$ for $i \in \{1, 2, \ldots, \eta\} \backslash \{\kappa + 1\}$ and $\mathbf{u}$ in the clear and $[\mathbf{P}_{\kappa+1}]$.
4. It also computes $[\mathbf{z}] = [\mathbf{v}_\eta \oslash \mathbf{w}]$ from $[\mathbf{v}_\eta]$ and $\mathbf{w}$.
5. It sets $\mathsf{y} = [\langle \mathbf{z}, \mathbf{1}_d \rangle]$ and $\pi := (\{[\mathbf{v}_i], [\mathbf{P}_i]\}_{i\in[\eta]}, [\mathbf{z}])$.
6. Finally, B returns y to A if $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 0$ and (y, π) to A if $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$.

At the end of the game, B outputs what A outputs.

To finish the proof of the lemma, it suffices to show that B simulates $\mathsf{Game}_{3,\kappa,1}$ if $\mathbf{A}$ is full-rank and $\mathsf{Game}_{3,\kappa,2}$ otherwise, except for negligible events that happen with probability at most $n_1 d/p$. We first observe that the two games differ only in how $\{[\mathbf{M}_{\kappa+1,j}]\}_{j\in[n_1]}$ are sampled. Furthermore, by inspection, it can be seen that other terms in vk and responses to the oracle queries are simulated as in $\mathsf{Game}_{3,\kappa,1}$ in the above simulation. Therefore, in the following, we focus on $\{[\mathbf{M}_{\kappa+1,j}]\}_{j\in[n_1]}$ and show that they are distributed as in $\mathsf{Game}_{3,\kappa,1}$ if $\mathbf{A}$ is full-rank and as in $\mathsf{Game}_{3,\kappa,2}$ otherwise, except for probability $n_1 d/p$.

The case of $\mathrm{rank}(\mathbf{A}) = d$**.** In this case, for all j, $\mathbf{F}_j$ is distributed uniformly at random over $\mathbb{Z}_p^{d\times d}$ and thus so is $\mathbf{M}_{\kappa+1,j} = (\mathbf{F}_j\mathbf{E}^{-1})^\top$. Therefore, the distribution of $\{\mathbf{M}_{\kappa+1,j}\}_j$ corresponds to that of $\mathsf{Game}_{3,\kappa,1}$.

The case of $\mathrm{rank}(\mathbf{A}) = d - 1$**.** In this case, we implicitly set $\mathcal{V}_\kappa$ to be a space spanned by the columns of $\mathbf{A}$. We analyze the distribution of $\mathbf{M}_{\kappa+1,j}$ for each j.

- **The case of** $j \in [n_1]\backslash\{j_{\kappa+1}\}$. In this case, each column of $\mathbf{F}_j$ is a random vector sampled from $\mathcal{V}_\kappa$. Let us assume that $\mathrm{rank}(\mathbf{F}_j) = d-1$, which happens with probability at least $1 - d/p$. We then have that for each $j \in [n_1]\backslash\{j_{\kappa+1}\}$, $\mathbf{F}_j$ is a random matrix of rank $d-1$ whose image is $\mathcal{V}_\kappa$. This in turn implies that $\mathbf{M}_{\kappa+1,j} = (\mathbf{F}_j\mathbf{E}^{-1})^\top$ is a random matrix of rank $d-1$ subject to $\mathcal{V}_\kappa = \mathbf{M}_{\kappa+1,j}^\top \mathcal{U}_\kappa$, which corresponds to the distribution of $\mathbf{M}_{\kappa+1,j}$ in $\mathsf{Game}_{3,\kappa,2}$.
- **The case of** $j = j_{\kappa+1}$. In this case, $\mathbf{f}_{j_{\kappa+1}}^1, \ldots, \mathbf{f}_{j_{\kappa+1}}^{d-1}$ are random vectors sampled from $\mathcal{V}_\kappa$ and $\mathbf{f}_{j_{\kappa+1}}^d$ is a random vector chosen from $\mathbb{Z}_p^d$. Let us assume that $\mathbf{f}_{j_{\kappa+1}}^1, \ldots, \mathbf{f}_{j_{\kappa+1}}^{d-1}$ span the entire space $\mathcal{V}_\kappa$ and $\mathbf{f}_{j_{\kappa+1}}^d$ is not in $\mathcal{V}_\kappa$. These two events happen with probability at least $1-d/p$. Then, $\mathbf{F}_{j_{\kappa+1}}$ is a random full-rank matrix with the constraint that the first $d-1$ columns span the space $\mathcal{V}_\kappa$. This in turn implies that $\mathbf{M}_{\kappa+1,j_{\kappa+1}} = (\mathbf{F}_{j_{\kappa+1}}\mathbf{E}^{-1})^\top$ is a random matrix of rank d subject to $\mathcal{V}_\kappa = \mathbf{M}_{\kappa+1,j_{\kappa+1}}^\top \mathcal{U}_\kappa$, which corresponds to the distribution of $\mathbf{M}_{\kappa+1,j_{\kappa+1}}$ in $\mathsf{Game}_{3,\kappa,2}$.

  In the above, for each j, we exclude the event that happens with probability at most d/p. Taking the union bound for all $j \in [n_1]$, these events do not happen except for probability at most $n_1 d/p$.

This completes the proof of Lemma 7. □

Lemma 8. *For all $\kappa \in \{0, 1, \ldots, \eta-1\}$, there exists an adversary B whose advantage against the d-rank problem is $|\Pr[\mathsf{E}_{3,\kappa,2}] - \Pr[\mathsf{E}_{3,\kappa+1,1}]| - dn_2/p$.*

Proof. The proof is very similar to that of Lemma 7. We will avoid repeating the same argument here and highlight the difference. We describe an adversary B that uses A to break the d-rank problem with advantage at least $|\Pr[\mathsf{E}_{3,\kappa,2}] - \Pr[\mathsf{E}_{3,\kappa+1,1}]| - dn_1/p$. Given the problem instance $(\mathcal{G}, [\mathbf{A}])$, B simulates vk as follows.

1. B first samples $\mathcal{U}_0, \mathcal{U}_1, \ldots, \mathcal{U}_\kappa$, $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_\kappa$, K, $\mathbf{u}$, and $\mathbf{w}$ as in $\mathsf{Game}_{3,\kappa,2}$.
2. It samples $\{\mathbf{M}_{i,j}\}_{i,j}$ and $\{\mathbf{R}_{i,k}\}_{i,k}$ for all i, j, k except for $\{\mathbf{R}_{\kappa+1,k}\}_{k\in[n_2]}$ as in $\mathsf{Game}_{3,\kappa,2}$.
3. It simulates $\{[\mathbf{N}_{\kappa+1,k}]\}_{k\in[n_2]}$ so that its distribution corresponds to that of $\mathsf{Game}_{3,\kappa,2}$ when $\mathbf{A}$ is full-rank and that of $\mathsf{Game}_{3,\kappa+1,1}$ when $\mathbf{A}$ is of rank $d-1$. We treat the case of $\kappa = \eta-1$ separately, since $\mathcal{U}_\eta$ is a space that is spanned by the first $d-1$ unit vectors, rather than a random subspace of $\mathbb{Z}_p^d$ with dimension $d-1$.
   - For $\kappa < \eta-1$, this can be done similarly to the proof of Lemma 7, where $\mathcal{V}_\kappa$, $\mathcal{U}_{\kappa+1}$, and $\{\mathbf{R}_{\kappa+1,k}\}_k$ here play the role of $\mathcal{U}_\kappa$, $\mathcal{V}_\kappa$, and $\{\mathbf{M}_{\kappa+1,j}\}_j$ there. We sample $\{[\mathbf{R}_{\kappa+1,k}]\}_k$ and then set $\mathbf{N}_{\kappa+1,k} = \mathbf{R}_{\kappa+1,k}$.
   - In the case of $\kappa = \eta-1$, we set $\mathcal{U}'_\eta := \mathbf{B}\mathcal{U}_\eta$ so that $\mathcal{U}'_\eta$ is a random subspace of $\mathbb{Z}_p^d$ with dimension $d-1$. Then, the same simulation strategy of Lemma 7 works, where $\mathcal{V}_{\eta-1}$, $\mathcal{U}'_\eta$, and $\{\mathbf{N}_{\eta,k}\}_k$ here play the role of $\mathcal{U}_\kappa$, $\mathcal{V}_\kappa$, and $\{\mathbf{M}_{\kappa+1,j}\}_j$ there. Here, we directly sample $\{[\mathbf{N}_{i,k}]\}_{i,k}$ rather than first sampling $\{[\mathbf{R}_{i,k}]\}_{i,k}$ and then setting $[\mathbf{N}_{i,k}] = [\mathbf{R}_{i,k}\mathbf{B}^\top]$.
4. Finally, B sets $\mathsf{vk} := (\mathcal{G}, [\mathbf{u}], [\mathbf{w}], \{[\mathbf{M}_{i,j}]\}_{i,j}, \{[\mathbf{N}_{i,k}]\}_{i,k})$ and gives it to A.

Similarly to the proof of Lemma 7, B can answer an oracle query x made by A, since it knows $\{[\mathbf{N}_{\kappa+1,k}]\}_k$ and $(\mathbf{u}, \mathbf{w}, \{\mathbf{M}_{i,j}\}_{i,j}, \{\mathbf{N}_{i,k}\}_{i\neq\kappa+1,k})$ in the clear (i.e., not on the exponent). □

Lemma 9. *We have $\mathsf{Game}_{3,\eta,1} \equiv \mathsf{Game}_4$.*

Proof. We can see that the distribution of vk in $\mathsf{Game}_{3,\eta,1}$ is exactly the same as that output by SimGen. To show that the two games are equivalent, we show that the oracle responses to a query x made by A in two games are equivalent. We consider the case of $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 0$ and $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$.

The case of $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$: It is straightforward to see that $(\{[\mathbf{v}_i], [\mathbf{P}_i]\}_{i\in[\eta]})$ returned to A as a part of the proof π in response to the oracle query x are computed in equivalent ways in the two games. Therefore, it remains to show that the same holds for y and $[\mathbf{z}]$ returned to A. In $\mathsf{Game}_{3,\eta,1}$, $\mathbf{z}$ is computed as $\mathbf{z} = \mathbf{v}_\eta \oslash \mathbf{w}$ while in Game_4, $\mathbf{z}$ is computed as $\mathbf{z} = \sum_{i=1}^{d-1} b_i \mathbf{a}_i$.

We have

$$\begin{aligned}
\mathbf{z} &= \mathbf{v}_\eta \oslash \mathbf{w} \\
&= \left(\left(\prod_{i=1}^{\eta} \mathbf{P}_i \right)^\top \mathbf{u} \right) \oslash \mathbf{w} \\
&= \left(\mathbf{B} \cdot \left(\prod_{i=1}^{\eta} \mathbf{Q}_i \right)^\top \mathbf{u} \right) \oslash \mathbf{c} \\
&= (\mathbf{B}\mathbf{b}) \oslash \mathbf{w} \\
&= ([\mathbf{a}_1 \odot \mathbf{c}, \cdots, \mathbf{a}_{d-1} \odot \mathbf{c}, \mathbf{d}]\mathbf{b}) \oslash \mathbf{w} \\
&= [\mathbf{a}_1, \cdots, \mathbf{a}_{d-1}, \mathbf{d} \oslash \mathbf{c}]\mathbf{b} \\
&= b_d \cdot (\mathbf{d} \oslash \mathbf{c}) + \sum_{i=1}^{d-1} b_i \mathbf{a}_i. \qquad (54)
\end{aligned}$$

Since $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 1$, we have $b_d = 0$ by Lemma 6 and thus $\mathbf{z}$ computed as above equals to $\mathbf{z} = \sum_{i=1}^{d-1} b_i \mathbf{a}_i$ as desired. Furthermore, since y is computed as $\mathsf{y} = [\langle \mathbf{1}_d, \mathbf{z} \rangle]$ in both games, the view of the adversary in both games is the same as well.

The case of $\mathsf{F}_{\mathsf{SSM}}(K, \mathsf{x}) = 0$: In $\mathsf{Game}_{3,\eta,1}$, y is computed as $\mathsf{y} = [\langle \mathbf{1}_d, \mathbf{z} \rangle]$ while it is computed as $\left[b_d t + \left\langle \mathbf{1}_d, \sum_{i=1}^{d-1} b_i \mathbf{a}_i \right\rangle \right]$ in Game_4. We have

$$[\langle \mathbf{1}_d, \mathbf{z} \rangle] = \left[b_d \langle \mathbf{1}_d, (\mathbf{d} \oslash \mathbf{c}) \rangle + \left\langle \mathbf{1}_d, \sum_{i=1}^{d-1} b_i \mathbf{a}_i \right\rangle \right] = \left[b_d t + \left\langle \mathbf{1}_d, \sum_{i=1}^{d-1} b_i \mathbf{a}_i \right\rangle \right],$$

where we use Eq. (54) in the first equation and $t = \mathbf{d} \oslash \mathbf{c}$ when $\psi \in \mathcal{D}_0$ in the second equation. Therefore, the view of the adversary in both games is the same in this case as well.

This completes the proof of Lemma 9. □

This completes the proof of Theorem 14. □

Remark 7 (On using random encoding function)**.** *Note that for the statement of Theorem 12 to be meaningful, we need the underlying partitioning function* $\mathsf{F}_{\mathsf{SSM}}$ *to have a good lower bound for the quantity* γ_{min}*. In particular, we would like to use Theorem 6. However, to invoke the theorem,* Encode *should be chosen randomly from a family of hash functions and included in* vk *unlike previous constructions, where* Encode *is a deterministic function. We discuss that this change does cause any problem. First of all, it is straightforward to see that this change does not ruin correctness and unique provability, since these properties hold for any encoding function* Encode*. In addition, this change does not harm the pseudorandomness. The only change we have to add to the statement of Theorem 12 is to modify the the inequality* $\epsilon_{\mathsf{B}} + 2\eta\epsilon_{\mathsf{B}'} \geq \gamma_{\mathsf{min}}\epsilon_{\mathsf{A}}/6|\Sigma|^{\eta'} - \eta d(n_1+n_2)/p$ *to be* $\epsilon_{\mathsf{B}} + 2\eta\epsilon_{\mathsf{B}'} \geq \gamma_{\mathsf{min}}\epsilon_{\mathsf{A}}/6|\Sigma|^{\eta'} - \eta d(n_1+n_2)/p - p_{c,\ell,\Sigma,n}$*, where* $p_{c,\ell,\Sigma,n}$ *is defined as in Lemma 5 and it accounts for the case where* Encode *chosen from the family of hash functions does not make corresponding* $\mathsf{F}_{\mathsf{SSM}}$ *a partitioning function with the desired parameters, due to the lack of small triple overlap property. We can make* $p_{c,\ell,\Sigma,n}$ *smaller than* $2^{-\ell}$ *by setting the parameters as in Theorem 12.*

7.5 Comparison

Here, we discuss our new VRF scheme constructed in Sec. 7.3 and compare it with previous schemes. We refer to Table 2 for the overview. For comparison, we focus on the schemes that achieve "all the desired properties" [HJ16]. Namely, we require the construction to have exponential-sized input space, adaptive security (i.e., both evaluation queries and the challenge query can be made adaptively), and security under non-interactive assumption. Here, we narrow the focus further and discuss constructions that are proven secure under a static assumption. We therefore do not include constructions that are proven secure under q-type assumptions [BMR10, HW10, Jag15, Yam17, Kat17, Nie21] or even stronger assumptions [JN19, JKN21] in the table. However, we mention that our construction achieves asymptotic efficiency that matches that of [Kat17], which is based on q-type assumptions. We also do not include the construction of VRFs from general assumptions that are quite inefficient [Bit17, GHKW17]. As we can see from the table, we achieve the best parameter size and reduction costs at the same time.

Since our construction is based on that of Kohl [Koh19], we compare our construction with hers in detail. Recall that there are two constructions of VRF in [Koh19] similarly to ours: one based on binary error correcting codes and the other based on error correcting codes with polynomial-size alphabets. Our improvement is based on two orthogonal ideas explained below:

- The first idea is to use 3-wise independent hash function as an error correcting codes. On the other hand, [Koh19] uses explicit constructions of error correcting codes both in binary and polynomial-sized alphabet cases. While the usage of 3-wise independent hash function requires that the description of the function should be included in the verification key, its description size is much smaller than other part of the verification key and can be ignored asymptotically. Simply replacing the underlying code in her construction with our code already leads to the following improvement.
  - In the polynomial-size alphabet setting, the usage of our code reduces the overall verification key size. The reduction in the verification key size is attributed to the reduction in the alphabet size $|\Sigma|$ of the code. Recall that in her construction, the verification key size is $\tilde{\Theta}(n|\Sigma|)$. We have $|\Sigma| = \ell^{\nu}$ for an arbitrarily chosen constant $\nu > 0$, while $|\Sigma| > \ell$ in her case.
  - Both in polynomial-size alphabet and binary alphabet settings, we can use our fine-tuned analysis of γ_{min} using the small triple overlap property. This leads to better reduction costs. In addition, in the polynomial-size alphabet setting, we can improve the reduction cost further, due to larger relative distance c of the our code. The reason why larger relative distance leads to better reduction cost is a bit technical. We refer to Remark 5 for detail.
- In addition, we alter the algebraic structure of the construction to significantly reduce the asymptotic size of the verification key. In her construction, she introduces a matrix of group elements for each combination of indices and alphabets, which leads to verification size $\tilde{\Theta}(n|\Sigma|)$. In contrast, we introduce two groups of matrices in the verification key and define another set of matrices by the combination of them. Then, each combination of the matrices is associated with the combination of indices and alphabets. This approach reduces the size of the verification key to approximately $\sqrt{\tilde{\Theta}(n|\Sigma|)}$. This idea for reducing the verification key size can be combined with the idea of using the 3-wise independent hash function in the first item.

Table 2: Comparison of VRF Schemes with All The Desired Properties Based on Standard Assumptions.

Schemes	$\lvert\mathsf{vk}\rvert$ (# of $\mathbb{G}$)	$\lvert\pi\rvert$ (# of $\mathbb{G}$)	Reduction Cost
[HJ16]	$O(\lambda)$	$O(\lambda)$	$\epsilon^{1+\mu}/\lambda Q^{\mu}$
[Ros18]	$O(\lambda)$	$O(\lambda)$	$\epsilon^{1+\mu}/\lambda Q^{\mu}$
[Koh19] (binary)	$\omega(\lambda \log \lambda)$	$\omega(\log \lambda)$	$\epsilon^{1+\mu}/\omega(\log \lambda) Q^{\mu}$
[Koh19] (polynomial)	$\omega(\lambda^{2+2\nu})$	$\omega(1)$	$\epsilon^{2+1/\nu}/\omega(1)\lambda^{1+\nu} Q^{1+1/\nu}$
Sec. 7.3 (binary)	$\omega(\sqrt{\lambda \log \lambda})$	$\omega(\log \lambda)$	$\epsilon^{1/2+\mu}/\omega(\log \lambda) Q^{\mu}$
Sec. 7.3 (polynomial)	$\omega(\lambda^{1/2+5\nu/2})$	$\omega(1)$	$\epsilon^{3/2}/\omega(1)\lambda^{\nu} Q$

We compare VRF schemes with all the desired properties proven secure under a static assumption. The constructions in the table are all proven secure under the d-LIN assumption. $\lvert\mathsf{vk}\rvert$ and $\lvert\pi\rvert$ represent the size of the verification keys and the size of the proofs, respectively. To measure $\lvert\mathsf{vk}\rvert$ and $\lvert\pi\rvert$, we count the number of group elements. Q and ϵ denote the number of evaluation queries and the advantage, respectively. $\mathsf{poly}(\lambda)$ represents fixed polynomial that does not depend on Q and ϵ. To measure the reduction cost, we show the advantage of the algorithm that solves the problem constructed from the adversary against the corresponding VRF scheme. We measure the reduction cost by employing the technique of Bellare and Ristenpart [BR09] for all the prior scheme and use our fine-tuned analysis for our schemes. In the table, μ and ν are arbitrary constants with $\mu > 1$ and $0 < \nu \leq 1$, respectively.

8 Computing $\widetilde{\gamma}(\vec{\mathsf{x}})$ Efficiently for Waters Hash $\mathsf{F}_{\mathsf{Wat}}$

In this section, we complete the proof of Theorem 2, showing how to efficiently compute the approximation $\widetilde{\gamma}(\vec{\mathsf{x}})$ for the partitioning function $\mathsf{F}_{\mathsf{Wat}}$ used by Waters [Wat05]. Although we can naively compute $\widetilde{\gamma}(\vec{\mathsf{x}})$ in time $\mathsf{poly}(Q, 1/\epsilon)$, such a large runtime defeats the purpose of a tighter advantage loss we obtained when compared to the security proof by Bellare and Ristenpart [BR09]. The main result of this section is to show how to compute $\widetilde{\gamma}(\vec{\mathsf{x}})$ in time $Q \cdot \mathsf{poly}(\lambda)$, which is of the same order as the reduction runtime of Bellare and Ristenpart. For our analysis, we use the powerful machinery of *generating functions*, which is a standard tool in enumerative combinatorics. The reason why we use the tool is that this greatly simplifies our analysis.

8.1 Preliminaries on Generating Functions

Here, we introduce necessary backgrounds for our purpose. For more information on the subject, we refer to [Wil05]. A generating function corresponding to a sequence $\{a_i \in \mathbb{R}\}_{i \in \mathbb{Z}_{\geq 0}}$ is a formal power series $f(\mathsf{Z}) = \sum_{i=0}^{\infty} a_i \mathsf{Z}^i$, where Z is indeterminate. When $a_i = 0$ for all $i > n$, we denote $f(\mathsf{Z}) = \sum_{i=0}^{n} a_i \mathsf{Z}^i$. We regard a generating function as an element in the formal power series ring $\mathbb{R}[[\mathsf{Z}]]$ and thus we can define the addition and multiplication of the generating functions. For $j \in \mathbb{Z}_{\geq 0}$ and $f(\mathsf{Z}) \in \mathbb{R}[[\mathsf{Z}]]$, by the symbol $[\mathsf{Z}^j] f(\mathsf{Z})$ we mean the coefficient of Z^j in the series. More explicitly, for $f(\mathsf{Z}) = \sum_{i=0}^{\infty} a_i \mathsf{Z}^i$, $[\mathsf{Z}^j] f(\mathsf{Z}) = a_j$. We can see that for $n \in \mathbb{N}$, $f(\mathsf{Z}) = \sum_{i=0}^{d_1} a_i \mathsf{Z}^i$, and $g(\mathsf{Z}) = \sum_{i=0}^{d_2} b_i \mathsf{Z}^i$ with $d_1 \leq n$, we have

$$[\mathsf{Z}^n](f(\mathsf{Z}) \cdot g(\mathsf{Z})) = \sum_{i=0}^{d_1} [\mathsf{Z}^i] f(\mathsf{Z}) \cdot [\mathsf{Z}^{n-i}] g(\mathsf{Z}). \tag{55}$$

When $f(\mathsf{Z}) = \sum_{i=0}^{\infty} a_i \mathsf{Z}^i$ has a multiplicative inverse, we denote it by $f(\mathsf{Z})^{-1}$. It can be easily checked that we have $(1 - \mathsf{Z})^{-1} = \sum_{i=0}^{\infty} \mathsf{Z}^i = 1 + \mathsf{Z} + \mathsf{Z}^2 + \cdots$. Furthermore, it is known that the

following equation holds:

$$(1-\mathsf{Z})^{-\ell} = \sum_{j=0}^{\infty} \binom{\ell+j-1}{\ell-1} \mathsf{Z}^j. \tag{56}$$

8.2 Combinatorial Lemmas

Here, we prove several important lemmas that are used for the analysis of our main algorithm in Section 8.3. We first introduce function $R_{N,\ell} : \mathbb{Z} \to \mathbb{Z}$ for $(N, \ell) \in \mathbb{N}^2$ defined as

$$R_{N,\ell}(\alpha) = \# \left\{ (K_j)_{j\in[\ell]} \in [0,N]^\ell \ : \ \sum_{j\in[\ell]} K_j = \alpha \right\}.$$

Lemma 10. *For all $N, \ell, \alpha \in \mathbb{N}$, we have $R_{N,\ell}(\alpha) = R_{N,\ell}(\ell N - \alpha)$.*

Proof. This can be seen by observing that the map $(K_1, \ldots, K_\ell) \mapsto (N-K_1, \ldots, N-K_\ell)$ gives a bijection between the sets $\left\{(K_j)_{j\in[\ell]} \in [0,N]^\ell : \sum_{j\in[\ell]} K_j = \alpha\right\}$ and $\left\{(K_j)_{j\in[\ell]} \in [0,N]^\ell : \sum_{j\in[\ell]} K_j = \ell N - \alpha\right\}$. □

A similar statement to the following lemma appears in [Wil05] in the form of exercise with a slightly different formulation. We provide the proof here for completeness. The lemma says that we can compute $R_{N,\ell}(\alpha)$ in time polylogarithmic in N. Looking ahead, this polylogarithmic efficiency is crucial when we use the lemma in Theorem 15.

Lemma 11 (Adapted from Exercise 10(g) of Section 1 in [Wil05])**.** *For $0 \le \alpha \le N\ell$, we have*

$$R_{N,\ell}(\alpha) = \sum_{i=0}^{\lfloor \frac{\alpha}{N+1} \rfloor} (-1)^i \binom{\ell}{i} \binom{\ell+\alpha-(N+1)i-1}{\ell-1}. \tag{57}$$

Furthermore, $R_{N,\ell}(\alpha)$ can be computed in time $O(\ell^2) \cdot \mathsf{poly}(\log N, \log \ell)$.

Proof. We first claim that

$$R_{N,\ell}(\alpha) = [\mathsf{Z}^\alpha](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^\ell$$

holds. This can be seen by observing that $(1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^\ell = \sum_{K_1,\ldots,K_\ell\in[0,N]} \mathsf{Z}^{K_1+\cdots+K_\ell}$. We then have

$$\begin{aligned}
R_{N,\ell}(\alpha) &= [\mathsf{Z}^\alpha](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^\ell \\
&= [\mathsf{Z}^\alpha]\left((1-\mathsf{Z}^{N+1})^\ell \cdot (1-\mathsf{Z})^{-\ell}\right) \\
&= [\mathsf{Z}^\alpha]\left(\left(\sum_{i=0}^{\ell}(-1)^i\binom{\ell}{i}\mathsf{Z}^{(N+1)i}\right)\cdot\left(\sum_{j=0}^{\infty}\binom{\ell+j-1}{\ell-1}\mathsf{Z}^j\right)\right) \\
&= [\mathsf{Z}^\alpha]\left(\sum_{i=0}^{\ell}\sum_{j=0}^{\infty}(-1)^i\binom{\ell}{i}\binom{\ell+j-1}{\ell-1}\mathsf{Z}^{(N+1)i+j}\right) \\
&= \sum_{i=0}^{\lfloor \frac{\alpha}{N+1} \rfloor}(-1)^i\binom{\ell}{i}\binom{\ell+\alpha-(N+1)i-1}{\ell-1},
\end{aligned}$$

where we use $(1-\mathsf{Z})(1+\mathsf{Z}+\mathsf{Z}^2+\cdots)=1$ in the first equation and binomial theorem and Eq. (56) in the third equation. To show that the latter part of the lemma holds, it suffices to note that there are at most ℓ terms in the summation and each of the binomial coefficient can be computed by $O(\ell)$ multiplications and a single division by using the formula $\binom{m}{n} = m(m-1)\cdots(m-n+1)/n!$. This completes the proof of the lemma. □

Lemma 12. *Let ℓ_1 and ℓ_2 be integers with $\ell_1 \leq \ell_2$. We have*

$$\sum_{\alpha=0}^{\ell_1 N} R_{N,\ell_1}(\alpha) R_{N,\ell_2}(\alpha) = R_{N,\ell_1+\ell_2}(\ell_2 N).$$

Proof. We have

$$\begin{aligned}
\sum_{\alpha=0}^{\ell_1 N} R_{N,\ell_1}(\alpha) R_{N,\ell_2}(\alpha) &= \sum_{\alpha=0}^{\ell_1 N} R_{N,\ell_1}(\alpha) R_{N,\ell_2}(\ell_2 N - \alpha) \\
&= \sum_{\alpha=0}^{\ell_1 N} [\mathsf{Z}^\alpha](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^{\ell_1} \cdot [\mathsf{Z}^{\ell_2 N-\alpha}](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^{\ell_2} \\
&= [\mathsf{Z}^{\ell_2 N}](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^{\ell_1+\ell_2} \\
&= R_{N,\ell_1+\ell_2}(\ell_2 N),
\end{aligned}$$

where we use Lemma 10 in the first equation, $R_{N,\ell}(\alpha) = [\mathsf{Z}^\alpha](1+\mathsf{Z}+\mathsf{Z}^2+\cdots+\mathsf{Z}^N)^\ell$ in the second and fourth equations, and Eq. (55) in the third equation. This completes the proof. □

8.3 An Efficient Algorithm $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}$ for Computing $\tilde{\gamma}(\vec{\mathsf{x}})$

With the preparation out of the way, we are now ready to state our main theorem, establishing the fact that $\tilde{\gamma}(\vec{\mathsf{x}})$ can be computed efficiently in time $Q \cdot \mathsf{poly}(\lambda)$.

Theorem 15. *Let the parameters $(Q, \epsilon, \ell, \vec{\mathsf{x}})$ be defined as in Theorem 2. Then, there exists an algorithm $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}(\lambda, Q, \epsilon, \vec{\mathsf{x}})$ that computes the function $\widetilde{\gamma}(\vec{\mathsf{x}}) = \Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j\in[Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$ in time $O(Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda)$. In particular, when $\ell = \mathsf{poly}(\lambda)$, it computes $\widetilde{\gamma}(\vec{\mathsf{x}})$ in time $Q \cdot \mathsf{poly}(\lambda)$.*

Proof. We first show the description of the algorithm $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}$ and analyze its running time. We postpone the explanation on the reason why $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}$ correctly computes $\widetilde{\gamma}(\vec{\mathsf{x}})$ by the procedure.

Now, we show the procedure of $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}$. It first computes $N = \lfloor \sqrt{3} \cdot Q/\sqrt{\epsilon} \rfloor$ according to the conditions of Theorem 2. Next, it computes $\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$ for all $j \in [Q]$ as follows.

1. It first defines $S := \{i \in [\ell] : \mathsf{x}^*_i = 1\}$ and $T := \{i \in [\ell] : \mathsf{x}^{(j)}_i = 1\}$. It sets $S' = S \setminus (S \cap T)$ and $T' = T \setminus (S \cap T)$.
2. It sets $\ell_1 = \min(|S'|, |T'|)$ and $\ell_2 = \max(|S'|, |T'|)$. Notice that $\ell_1 + \ell_2 \leq \ell$.
3. If $\ell_1 = 0$ holds, it computes

$$\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] = \begin{cases} 1/(\ell N+1)(N+1)^{\ell_2} & \text{if } \ell_1 = 0 \\ R_{N,\ell_1+\ell_2}(\ell_2 N)/(\ell N+1)(N+1)^{\ell_1+\ell_2} & \text{otherwise.} \end{cases} \tag{58}$$

where $R_{N,\ell_1+\ell_2}(\ell_2 N)$ is computed using Eq. (57) in the above. We postpone the proof of Eq. (58) for the time being.

It finally computes $\widetilde{\gamma}(\vec{\mathsf{x}}) = \Pr[\mathsf{E}(\mathsf{x}^*)] - \sum_{j\in[Q]} \Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$. Recall that $\mathsf{E}(\mathsf{x}^*) = 1/(\ell N + 1)$.

Let us evaluate the running time of $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}$. It can compute N in $O(1)$ arithmetic operations. Now we focus on the computation of $\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$ for each $j \in [Q]$. The computation in the first and second steps and the third step for the case of $\ell_1 = 0$ requires (ℓ) arithmetic operations. Due to Lemma 11, computing $R_{N,\ell_1+\ell_2}(\ell_2 N)$ takes $O((\ell_1+\ell_2)^2) \cdot \mathsf{poly}(\log N, \log(\ell_1+\ell_2))$ time. Thus, since $\ell_1+\ell_2 \leq \ell$, computing $\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})]$ for all $j \in [Q]$ takes $O(Q \cdot \ell^2) \cdot \mathsf{poly}(\log N, \log \ell)$ time. The computation of $\widetilde{\gamma}(\vec{\mathsf{x}})$ at the end of the procedure requires $O(\ell)$ arithmetic operation. Thus, $\mathsf{Alg}_{\mathsf{Wat},\widetilde{\gamma}}$ terminates in time $O(Q \cdot \ell^2) \cdot \mathsf{poly}(\log N, \log \ell)$. Since Q and ℓ are polynomially bounded and ϵ is noticeable, this concludes that it terminates in time $Q \cdot \mathsf{poly}(\lambda)$.

To finish this proof, it remains to show Eq. (58). We first show the case of $\ell_1 = 0$. Without loss of generality, we assume $|S'| = 0$, namely $S \subset T$. Then, $\ell_2 = |T'|$ holds. We have

$$\begin{aligned}
\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] &= \Pr\left[K_0 + \sum_{i\in S} K_i = 0 \ \wedge \ K_0 + \sum_{i\in T} K_i = 0\right] \\
&= \Pr\left[K_0 + \sum_{i\in S} K_i = 0 \ \wedge \ \sum_{i\in T'} K_i = 0\right] \\
&= \Pr\left[K_0 + \sum_{i\in S} K_i = 0\right] \cdot \Pr\left[\sum_{i\in T'} K_i = 0\right] \\
&= \frac{1}{(\ell N+1)(N+1)^{\ell_2}}
\end{aligned}$$

where the third equality follows from $S \cap T' = \emptyset$. Next, we consider the case of $\ell_1 > 0$. We have the following:

$$\begin{aligned}
\Pr[\mathsf{E}(\mathsf{x}^*) \wedge \mathsf{E}(\mathsf{x}^{(j)})] &= \Pr\left[K_0 + \sum_{i\in S} K_i = 0 \ \wedge \ K_0 + \sum_{i\in T} K_i = 0\right] \\
&= \Pr\left[K_0 + \sum_{i\in S} K_i = 0 \ \wedge \ \sum_{i\in S'} K_i = \sum_{i\in T'} K_i\right] \\
&= \sum_{z=0}^{\ell N} \Pr[K_0 = -z] \cdot \Pr\left[\sum_{i\in S} K_i = -K_0 \ \wedge \ \sum_{i\in S'} K_i = \sum_{i\in T'} K_i \,\middle|\, K_0 = -z\right] \\
&= \frac{1}{\ell N+1} \cdot \sum_{z=0}^{\ell N} \Pr\left[\sum_{i\in S} K_i = z \ \wedge \ \sum_{i\in S'} K_i = \sum_{i\in T'} K_i\right] \\
&= \frac{1}{\ell N+1} \cdot \Pr\left[\sum_{i\in S'} K_i = \sum_{i\in T'} K_i\right].
\end{aligned}$$

Now, without loss of generality, we assume $|S'| \leq |T'|$, namely, $|S'| = \ell_1$ and $|T'| = \ell_2$. Then, we obtain

$$\begin{aligned}
&\Pr\left[\sum_{i\in S'} K_i = \sum_{i\in T'} K_i\right] \\
&= \sum_{z=0}^{\ell_1 N} \Pr\left[\sum_{i\in S'} K_i = \sum_{i\in T'} K_i = z\right]
\end{aligned}$$

$$= \frac{1}{(N+1)^{\ell_1+\ell_2}} \cdot \sum_{z=0}^{\ell_1 N} \# \left\{ K_i \in [0,N] \text{ for } i \in S' \ : \sum_{i \in S'} K_i = z \right\} \cdot \# \left\{ K_i \in [0,N] \text{ for } i \in T' \ : \sum_{i \in T'} K_i = z \right\}$$
$$= \frac{1}{(N+1)^{\ell_1+\ell_2}} \cdot \sum_{z=0}^{\ell_1 N} R_{N,\ell_1}(z) \cdot R_{N,\ell_2}(z)$$
$$= \frac{1}{(\ell N+1)(N+1)^{\ell_1+\ell_2}} \cdot R_{N,\ell_1+\ell_2}(\ell_2 N)$$

where the first equality follows from the facts that $\ell_1 \leq \ell_2$ and $\sum_{i \in S'} K_i \leq \ell_1 N$, the second equality follows from the fact that K_i for $i \in S' \cup T'$ is chosen uniformly at random from $[0,N]$, the third equality follows from the definition of $R_{N,\ell}$, and the final equality follows from Lemma 12. This completes the proof. □

Acknowledgement

We thank Weidan Ji for pointing out an error in the proof of Theorem 1 in the previous version. The first and the last author were partly supported by JST AIP Acceleration Research JPMJCR22U5. The third author is partially supported by JSPS KAKENHI Grant Number JP21K17700. The last author was supported by JST CREST Grant Number JPMJCR22M1.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Heidelberg, August 2010.

[ACF09] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 554–571. Springer, Heidelberg, April 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, July 2014.

[AFL17] Daniel Apon, Xiong Fan, and Feng-Hao Liu. Vector encoding over lattices and its applications. Cryptology ePrint Archive, Report 2017/455, 2017. https://eprint.iacr.org/2017/455.

[AHY15] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. A framework for identity-based encryption with almost tight security. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 521–549. Springer, Heidelberg, November / December 2015.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[ALWW21] Parhat Abla, Feng-Hao Liu, Han Wang, and Zhedong Wang. Ring-based identity based encryption - asymptotically shorter MPK and tighter security. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 157–187. Springer, Heidelberg, November 2021.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, August 2004.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th FOCS*, pages 647–657. IEEE Computer Society Press, October 2007.

[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.

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 567–594. Springer, Heidelberg, November 2017.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014.

[BL16] Xavier Boyen and Qinyi Li. Towards tightly secure lattice short signature and id-based encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 404–434. Springer, Heidelberg, December 2016.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 131–140. ACM Press, October 2010.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 2005*, pages 320–329. ACM Press, November 2005.

[Bon36] C. Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936.

[Bon01] Dan Boneh. Simplified OAEP for the RSA and Rabin functions. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 275–291. Springer, Heidelberg, August 2001.

[Boy10] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, Heidelberg, May 2010.

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 407–424. Springer, Heidelberg, April 2009.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384. Springer, Heidelberg, August 2016.

[CGW17] Jie Chen, Junqing Gong, and Jian Weng. Tightly secure IBE under constant-size master public key. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 207–231. Springer, Heidelberg, March 2017.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012.

[CLL+13] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 122–140. Springer, Heidelberg, May 2013.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, December 2001.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

[DKPW12] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Heidelberg, April 2012.

[Dod03] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 1–17. Springer, Heidelberg, January 2003.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 513–530. Springer, Heidelberg, August 2013.

[GDCC16] Junqing Gong, Xiaolei Dong, Jie Chen, and Zhenfu Cao. Efficient IBE with tight reduction to standard assumption in the multi-challenge setting. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 624–654. Springer, Heidelberg, December 2016.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, Heidelberg, May / June 2006.

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 537–566. Springer, Heidelberg, November 2017.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[HJ16] Dennis Hofheinz and Tibor Jager. Verifiable random functions from standard assumptions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 336–362. Springer, Heidelberg, January 2016.

[HJK12] Dennis Hofheinz, Tibor Jager, and Edward Knapp. Waters signatures with optimal security reduction. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 66–83. Springer, Heidelberg, May 2012.

[HK08] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Heidelberg, August 2008.

[HW09] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 333–350. Springer, Heidelberg, April 2009.

[HW10] Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 656–672. Springer, Heidelberg, May / June 2010.

[Jag15] Tibor Jager. Verifiable random functions from weaker assumptions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 121–143. Springer, Heidelberg, March 2015.

[JKN21] Tibor Jager, Rafael Kurek, and David Niehues. Efficient adaptively-secure IB-KEMs and VRFs via near-collision resistance. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 596–626. Springer, Heidelberg, May 2021.

[JN19] Tibor Jager and David Niehues. On the real-world instantiability of admissible hash functions and efficient verifiable random functions. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 303–332. Springer, Heidelberg, August 2019.

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.

[Kat17] Shuichi Katsumata. On the untapped potential of encoding predicates by arithmetic circuits and their applications. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 95–125. Springer, Heidelberg, December 2017.

[Koh19] Lisa Kohl. Hunting and gathering - verifiable random functions from standard assumptions with short proofs. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 408–437. Springer, Heidelberg, April 2019.

[KPC+11] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 7–26. Springer, Heidelberg, May 2011.

[KTY23] Shuichi Katsumata, Toi Tomita, and Shota Yamada. Direct computation of branching programs and its applications to more efficient lattice-based cryptography. *Des. Codes Cryptogr.*, 91(2):391–431, 2023.

[KY16] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Heidelberg, December 2016.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, April 2012.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, August 2002.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

[Nie21] David Niehues. Verifiable random functions with optimal tightness. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 61–91. Springer, Heidelberg, May 2021.

[RCS12] Somindu C. Ramanna, Sanjit Chatterjee, and Palash Sarkar. Variants of waters' dual system primitives using asymmetric pairings - (extended abstract). In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 298–315. Springer, Heidelberg, May 2012.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[Ros18] Razvan Rosie. Adaptive-secure VRFs with shorter keys from static assumptions. In Jan Camenisch and Panos Papadimitratos, editors, *CANS 18*, volume 11124 of *LNCS*, pages 440–459. Springer, Heidelberg, September / October 2018.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, Japan, January 2000.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

[Wil05] Herbert S. Wilf. *Generatingfunctionology: Third Edition*. A K Peters/CRC Press, 2005.

[Yam16] Shota Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 32–62. Springer, Heidelberg, May 2016.

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 161–193. Springer, Heidelberg, August 2017.

[ZCZ16] Jiang Zhang, Yu Chen, and Zhenfeng Zhang. Programmable hash functions from lattices: Short signatures and IBEs with small key sizes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 303–332. Springer, Heidelberg, August 2016.

A Details on Application to Waters IBE

In this section, we provide omitted details from Sec. 6.2. Concretely, we provide the definition of the DBDH assumption, the description of the Waters IBE scheme [Wat05], partitioning based reduction for the scheme, and the proof of Theorem 8.

A.1 Preliminalies

Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ be pairing parameters where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of prime order p and e is a non-degenerate and efficiently computable bilinear map. We denote the set of non-identity elements in $\mathbb{G}_i$ by $\mathbb{G}_i^*$ for any $i \in \{1, 2, T\}$.

We now recall tbe definition of decisional bilinear Diffie-Hellman (DBDH) problem.

Definition 17 (DBDH Problem)**.** *For pairing parameters* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ *and a PPT algorithm* A*, the advantage of* A *for the DBDH problem is defined by*

$$\mathsf{Adv}^{\mathrm{dbdh}}(\mathsf{A}) = \left|\Pr[\mathsf{A}(\{g_i, g_i^a, g_i^b, g_i^c\}_{i\in\{1,2\}}, e(g_1,g_2)^{abc}) = 1] - \Pr[\mathsf{A}(\{g_i, g_i^a, g_i^b, g_i^c\}_{i\in\{1,2\}}, e(g_1,g_2)^{z}) = 1]\right|$$

where $g_1 \xleftarrow{\$} \mathbb{G}_1^*$, $g_2 \xleftarrow{\$} \mathbb{G}_2^*$, *and* $a, b, c, z \xleftarrow{\$} \mathbb{Z}_p$. *We say that the DBDH problem is hard if* $\mathsf{Adv}^{\mathrm{dbdh}}(\mathsf{A})$ *is negligible for all PPT algorithm* A.

A.2 Waters IBE and Partitioning-Based Reduction for the Scheme

In Fig. 2, we provide the description of Waters IBE scheme. Our description of Waters IBE scheme is different from the original one [Wat05] in that we use asymmetric pairings, which allows us to have a better efficiency. It is also different from the one by Bellare and Ristenpart [BR09], who uses asymmetric pairings. The difference from the latter is that we make sure that all the ciphertext components except for the message carrying part reside in $\mathbb{G}_1$, whereas in their description, they are mix of $\mathbb{G}_1$ and $\mathbb{G}_2$ elements. By using asymmetric pairing that minimizes the description size of $\mathbb{G}_1$ elements, we can minimize the ciphertext size.

$\mathsf{Setup}(1^\lambda)$

$(g_1, g_2) \xleftarrow{\$} \mathbb{G}_1^* \times \mathbb{G}_2^*$,
$(a, b) \xleftarrow{\$} \mathbb{Z}_p^2$
$A_1 := g_1^a,\ B_2 := g_2^b$
$(u_0, u_1, \ldots, u_\ell) \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$
for $i \in [0, \ell]$ **do**
  $U_i := g_1^{u_i},\ U_i' := g_2^{u_i}$
$\mathbf{U} := (U_0, U_0', \ldots, U_\ell, U_\ell')$
$\mathsf{mpk} := (g_1, A_1, g_2, B_2, \mathbf{U})$
$\mathsf{msk} := g_2^{ab}$
return $(\mathsf{mpk}, \mathsf{msk})$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$

parse $(g_1, A_1, g_2, B_2, \mathbf{U}) \leftarrow \mathsf{mpk}$
parse $(U_0, U_0', \ldots, U_\ell, U_\ell') \leftarrow \mathbf{U}$
$r \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{sk}_{\mathsf{ID}}^{(1)} := \mathsf{msk} \cdot \left(U_0' \prod_{i:\mathsf{ID}_i=1} U_i' \right)^r$
$\mathsf{sk}_{\mathsf{ID}}^{(2)} := g_2^r$
$\mathsf{sk}_{\mathsf{ID}} \leftarrow (\mathsf{sk}_{\mathsf{ID}}^{(1)}, \mathsf{sk}_{\mathsf{ID}}^{(2)})$
return $\mathsf{sk}_{\mathsf{ID}}$

$\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$

parse $(g_1, A_1, g_2, B_2, \mathbf{U}) \leftarrow \mathsf{mpk}$
parse $(U_0, U_0', \ldots, U_\ell, U_\ell') \leftarrow \mathbf{U}$
$c \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{ct}^{(1)} := e(A_1, B_2)^c \cdot \mathsf{M}$
$\mathsf{ct}^{(2)} := g_1^c$
$\mathsf{ct}^{(3)} := \left(U_0 \prod_{i:\mathsf{ID}_i=1} U_i \right)^c$
$\mathsf{ct} \leftarrow (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)}, \mathsf{ct}^{(3)})$
return ct

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct})$

if ct is not in a valid form
  return $\perp$
parse $(\mathsf{sk}_{\mathsf{ID}}^{(1)}, \mathsf{sk}_{\mathsf{ID}}^{(2)}) \leftarrow \mathsf{sk}_{\mathsf{ID}}$
parse $(\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)}, \mathsf{ct}^{(3)}) \leftarrow \mathsf{ct}$
$\mathsf{M} \leftarrow \mathsf{ct}^{(1)} \cdot \frac{e(\mathsf{ct}^{(3)}, \mathsf{sk}_{\mathsf{ID}}^{(2)})}{e(\mathsf{ct}^{(2)}, \mathsf{sk}_{\mathsf{ID}}^{(1)})}$
return M

Figure 2: Waters IBE Scheme.

The following lemma claims that Waters IBE scheme admits a partitioning-based reduction from the DBDH problem with respect to $\mathsf{F}_{\mathsf{Wat}}$. Essentially, the lemma is proven by Waters [Wat05] implicitly and here we translate his proof into our language of partitioning based reduction.

Lemma 13. *There is* $(0, 0, 0, 0)$*- partitioning-based reduction for the Waters IBE scheme from the DBDH problem with respect to* $\mathsf{F}_{\mathsf{Wat}}$.

$\mathsf{SimSetup}(\mathbf{X}, \psi)$

1 : **parse** $(X_i)_{i\in[0,\ell]} \leftarrow \mathbf{X} \in \mathbb{Z}_p^{\ell+1}$
2 : **parse** $((g_b, A_b, B_b, C_b)_{b\in\{0,1\}}, W) \leftarrow \psi$
3 : **for** $i \in [0,\ell]$ **do**
4 : $\quad Y_i \xleftarrow{\$} \mathbb{Z}_p$
5 : $\quad U_i := B_1^{X_i} g_1^{Y_i},\ U_i' := B_2^{X_i} g_2^{Y_i}$
6 : $\mathbf{Y} := (Y_0, Y_1, \ldots, Y_\ell)$
7 : $\mathbf{U} := (U_0, U_0', \ldots, U_\ell, U_\ell')$
8 : $\mathsf{mpk} := (g_1, A_1, g_2, B_2, \mathbf{U})$
9 : $\mathsf{td} := (\mathbf{Y}, \mathsf{mpk}, \psi)$
10 : **return** $(\mathsf{mpk}, \mathsf{td})$

$\mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID})$

1 : $r \xleftarrow{\$} \mathbb{Z}_p$
2 : $\mathsf{sk}_{\mathsf{ID}}^{(1)} := A_2^{-\frac{\mathsf{G}(\mathbf{Y},\mathsf{ID})}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \cdot \left(B_2^{\mathsf{G}(\mathbf{X},\mathsf{ID})} g_2^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}\right)^r$
3 : $\mathsf{sk}_{\mathsf{ID}}^{(2)} := g_2^r A_2^{-\frac{1}{\mathsf{G}(\mathbf{X},\mathsf{ID})}}$
4 : $\mathsf{sk}_{\mathsf{ID}} := (\mathsf{sk}_{\mathsf{ID}}^{(1)}, \mathsf{sk}_{\mathsf{ID}}^{(2)})$
5 : **return** $\mathsf{sk}_{\mathsf{ID}}$

$\mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}, \mathsf{M})$

1 : $\mathsf{ct}^{(1)} := W \cdot \mathsf{M}$
2 : $\mathsf{ct}^{(2)} := C_1, \quad$ and
3 : $\mathsf{ct}^{(3)} := C_1^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}$
4 : $\mathsf{ct} \leftarrow (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)}, \mathsf{ct}^{(3)})$
5 : **return** ct

Hard Distribution $\mathcal{D}_0$

1 : $(g_1, g_2) \xleftarrow{\$} \mathbb{G}_1^* \times \mathbb{G}_2^*$,
2 : $(a, b, c) \xleftarrow{\$} \mathbb{Z}_p^3$
3 : **for** $i \in \{1, 2\}$ **do**
4 : $\quad A_i := g_i^a,\ B_i := g_i^b,\ C_i := g_i^c$
5 : $W := e(g_1, g_2)^{abc}$
6 : $\psi_0 := ((g_b, A_b, B_b, C_b)_{b\in\{0,1\}}, W)$
7 : **return** ψ_0

Hard Distribution $\mathcal{D}_1$

1 : $(g_1, g_2) \xleftarrow{\$} \mathbb{G}_1^* \times \mathbb{G}_2^*$,
2 : $(a, b, c, z) \xleftarrow{\$} \mathbb{Z}_p^4$
3 : **for** $i \in \{1, 2\}$ **do**
4 : $\quad A_i := g_i^a,\ B_i := g_i^b,\ C_i := g_i^c$
5 : $W := e(g_1, g_2)^z$
6 : $\psi_1 := ((g_b, A_b, B_b, C_b)_{b\in\{0,1\}}, W)$
7 : **return** ψ_1

Figure 3: Algorithms used by the partitioning-based reduction for the Waters IBE scheme. G is a function $\mathsf{G} : \mathbb{Z}_p^{\ell+1} \times \{0,1\}^\ell \to \mathbb{Z}_p$ such that $\mathsf{G}(\mathbf{Z}, \mathsf{ID}) = Z_0 + \sum_{i:\mathsf{ID}_i=1} Z_i \mod p$ for $\mathbf{Z} = (Z_0, Z_1, \ldots, Z_\ell) \in \mathbb{Z}_p^{\ell+1}$ and $\mathsf{ID} \in \{0,1\}^\ell$.

Proof. Let us define a function $\mathsf{G} : \mathbb{Z}_p^{\ell+1} \times \{0,1\}^\ell \to \mathbb{Z}_p$ as $\mathsf{G}(\mathbf{Z}, \mathsf{ID}) = Z_0 + \sum_{i:\mathsf{ID}_i=1} Z_i \mod p$ for $\mathbf{Z} = (Z_0, Z_1, \ldots, Z_\ell) \in \mathbb{Z}_p^{\ell+1}$ and $\mathsf{ID} \in \{0,1\}^\ell$.

We construct three simulation algorithms $(\mathsf{SimSetup}, \mathsf{SimKeyGen}, \mathsf{SimEncrypt})$ in Fig. 3. It is easy to check that the running time of each algorithm is $\mathsf{poly}(\lambda)$ related to their counterpart real algorithm. Below, we show that these algorithm satisfies all properties of Def. 11.

Master public key simulatability. For $\psi \in \mathcal{D}_0 \cup \mathcal{D}_1$, (g_1, A_1, g_2, B_2) is uniformly distributed over $\mathbb{G}_1^2 \times \mathbb{G}_2^2$. Since $\mathbf{Y}$ is chosen uniformly at random from $\mathbb{Z}_p^{\ell+1}$, $\{U_i\}_{i\in[0,\ell]}$ and $\{U_i'\}_{i\in[0,\ell]}$ are uniformly distributed over $\mathbb{G}_1^{\ell+1}$ and $\mathbb{G}_2^{\ell+1}$, respectively. Thus, the distributions of $\mathbf{U}$ output by $\mathsf{SimSetup}$ and Setup are identical. Therefore, we obtain $\epsilon_{\mathsf{S}} = 0$.

Secret key simulatability. We now consider ID such that $\mathsf{F}_{\mathsf{Wat}}(\mathbf{X}, \mathsf{ID}) = 1$. Since $\mathsf{G}(\mathbf{X}, \mathsf{ID}) \neq 0$, $\mathsf{sk}_{\mathsf{ID}}^{(1)}$ and $\mathsf{sk}_{\mathsf{ID}}^{(2)}$ are computable. Let us fix any $\mathsf{mpk} = (g_1, A_1, g_2, B_2, \mathbf{U})$. Notice that there exists one $\mathsf{msk} \in S_{\mathsf{mpk}}$ for any mpk. Since $U_i = B_1^{X_i} g_1^{Y_i}$ and $U_i' = B_2^{X_i} g_2^{Y_i}$ holds for all $i \in [0,\ell]$,

$\mathsf{sk}_{\mathsf{ID}} = (\mathsf{sk}_{\mathsf{ID}}^{(1)}, \mathsf{sk}_{\mathsf{ID}}^{(2)})$ output by $\mathsf{SimKeyGen}$ satisfies

$$\begin{aligned}
&A_2^{-\frac{\mathsf{G}(\mathbf{Y},\mathsf{ID})}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \cdot \left(B_2^{\mathsf{G}(\mathbf{X},\mathsf{ID})} g_2^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}\right)^r \\
&= A_2^{-\frac{\mathsf{G}(\mathbf{Y},\mathsf{ID})}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \cdot \left(B_2^{\mathsf{G}(\mathbf{X},\mathsf{ID})} g_2^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}\right)^{\frac{a}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \cdot \left(B_2^{\mathsf{G}(\mathbf{X},\mathsf{ID})} g_2^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}\right)^{r-\frac{a}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \\
&= A_2^{-\frac{\mathsf{G}(\mathbf{Y},\mathsf{ID})}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \cdot B_2^a \cdot A_2^{\frac{\mathsf{G}(\mathbf{Y},\mathsf{ID})}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \cdot \left(U_0' \prod_{i:\mathsf{ID}_i=1} U_i'\right)^{r-\frac{a}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} \\
&= \mathsf{msk} \cdot \left(U_0' \prod_{i:\mathsf{ID}_i=1} U_i'\right)^{r-\frac{a}{\mathsf{G}(\mathbf{X},\mathsf{ID})}}
\end{aligned}$$

and

$$g_2^r A_2^{-\frac{1}{\mathsf{G}(\mathbf{X},\mathsf{ID})}} = g_2^{r-\frac{a}{\mathsf{G}(\mathbf{X},\mathsf{ID})}}.$$

Because r is chosen uniformly at random from $\mathbb{Z}_p$, $r - a/\mathsf{G}(\mathbf{X},\mathsf{ID})$ is uniformly distributed over $\mathbb{Z}_p$. Thus, the distributions of $\mathsf{sk}_{\mathsf{ID}}$ output by $\mathsf{SimKeyGen}$ and KeyGen are identical for all ID such that $\mathsf{F}_{\mathsf{Wat}}(\mathbf{X},\mathsf{ID}) = 1$. Therefore, $\epsilon_{\mathsf{K}} = 0$ holds.

Ciphertext simulatability. We consider ID such that $\mathsf{F}_{\mathsf{Wat}}(\mathbf{X},\mathsf{ID}) = 0$, namely, $\mathsf{G}(\mathbf{X},\mathsf{ID}) = 0$ holds. Let us fix any $(\mathsf{mpk},\mathsf{td})$ generated by $\mathsf{SimSetup}(\mathbf{X},\psi)$ where $\psi \xleftarrow{\$} \mathcal{D}_0$. For $\mathsf{ct}^{(3)}$, we have $\left(U_1 \prod_{i:\mathsf{ID}_i=1} U_i\right)^c = \left(B_1^{\mathsf{G}(\mathbf{X},\mathsf{ID})} g_1^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}\right)^c = C_1^{\mathsf{G}(\mathbf{Y},\mathsf{ID})}$. Then, since C_1 is uniformly distributed over $\mathbb{G}_1$, the distributions of $\mathsf{ct}^{(2)}$ and $\mathsf{ct}^{(3)}$ output by $\mathsf{SimEncrypt}$ and $\mathsf{Encrypt}$ are identical. When td is generated by $\mathcal{D}_0$, $W = e(g_1,g_2)^{abc} = e(A_1,B_2)^c$ holds. Thus, the distributions of $\mathsf{ct}^{(1)}$ output by $\mathsf{SimEncrypt}$ and $\mathsf{Encrypt}$ are also identical. Therefore, for all $\mathsf{ID} \in \{0,1\}^\ell$ such that $\mathsf{F}(K,\mathsf{ID}^*) = 0$ holds and td computed from $\psi \xleftarrow{\$} \mathcal{D}_0$, the distribution of ct generated by $\mathsf{SimEncrypt}$ and $\mathsf{Encrypt}$ are identical. Thus, we have $\epsilon_{\mathsf{E}} = 0$.

Ciphertext randomizability. We consider ID such that $\mathsf{F}_{\mathsf{Wat}}(\mathbf{X},\mathsf{ID}) = 0$ and $(\mathsf{mpk},\mathsf{td})$ generated by $\mathsf{SimSetup}(\mathbf{X},\psi)$ where $\psi \xleftarrow{\$} \mathcal{D}_1$. Since W is uniformly distributed over $\mathbb{G}_T$, $\mathsf{ct}^{(1)}$ generated by $\mathsf{SimEncrypt}$ is uniformly distributed over $\mathbb{G}_T$ independently of a message M to be encrypted. Moreover, $\mathsf{ct}^{(2)}$ and $\mathsf{ct}^{(3)}$ are produced irrelevantly to M in $\mathsf{SimEncrypt}$. Thus, for all $\mathsf{M},\mathsf{M}^* \in \mathcal{M}$, the distributions of ct generated by $\mathsf{SimEncrypt}(\mathsf{td},\mathsf{ID}^*,\mathsf{M})$ and $\mathsf{SimEncrypt}(\mathsf{td},\mathsf{ID}^*,\mathsf{M}^*)$ are identical. Therefore, we have $\epsilon_{\mathsf{R}} = 0$. This completes this proof. □

A.3 Proof for Theorem 8

The following theorem asserts the security of Waters IBE scheme. The proof of the theorem follows from Theorem 7 and Theorem 2 and Lemma 13 shown in the previous sections.

Theorem 16 (Restatement of Theorem 8)**.** *If there is an $(t_{\mathsf{A}},Q,\epsilon_{\mathsf{A}})$-adversary A against the $\mathsf{IND\text{-}CPA}$ security of the Waters IBE scheme, there is an adversary B that breaks the DBDH problem with advantage ϵ_{B} and t_{B} such that*

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}^{1.5}}{21Q\ell}, \quad t_{\mathsf{B}} = t_{\mathsf{A}} + O(Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda)$$

where $Q \leq p\sqrt{\epsilon_{\mathsf{A}}}/\ell\sqrt{3}$ and $\mathsf{poly}(\lambda)$ is roughly the overhead incurred by the running the simulated algorithms compared to the real $(\mathsf{Setup},\mathsf{KeyGen},\mathsf{Encrypt})$ algorithms.

Proof. By applying Theorem 2 and Lemma 13 to Theorem 7, we have

$$\begin{aligned} t_{\mathsf{B}} &= t_{\mathsf{A}} + O(Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda) + Q \cdot (\ell \cdot \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda)) \\ &= t_{\mathsf{A}} + O(Q \cdot \ell^2) \cdot \mathsf{poly}(\lambda), \\ \text{and } \epsilon_{\mathsf{B}} &\geq \frac{\gamma_{\mathsf{min}}}{3}\epsilon_{\mathsf{A}} > \frac{\epsilon_{\mathsf{A}}^{1.5}}{21Q\ell}. \end{aligned}$$

This completes the proof. □

A.4 A Variant of Waters IBE from the CBDH Assumption

Here, we briefly discuss the variant of Waters IBE that can be proven secure under the computational bilinear Diffie-Hellman (CBDH) assumption[15], which assumes that given $(g_i^a, g_i^b, g_i^c)_{i=1,2}$ with random a, b, c on a bilinear group, it is hard to compute $e(g_1, g_2)^{abc}$. The assumption is potentially strictly weaker than the DBDH assumption. We can base the security of Waters IBE on the CBDH assumption with slight modification. For simplicity, let us consider a variant with a single bit message space. In the modified scheme, we add a random string r whose length is the same as the binary length of $\mathbb{G}_T$ element to mpk and then mask the message $\mathsf{M} \in \{0,1\}$ by $\langle r, W\rangle \oplus \mathsf{M}$ in the ciphertext. Due to the Goldreich-Levin hardcore bit theorem [GL89], the term $\langle r, W\rangle$ is pseudorandom assuming the CBDH assumption. We can prove the security of this variant with very small change from the case of DBDH. Our improvement on the reduction cost can be applied to this variant as well.

B Details on Applications to Lattice IBEs

In this section, we provide omitted details from Sec. 6.3. Concretely, we provide backgrounds on lattices, the description of the ABB IBE scheme [ABB10a] and our variant d-extended ABB scheme, partitioning based reduction for the schemes, and the proofs of Theorems 9 and 10.

B.1 Preliminaries on Lattices

Distributions. For an integer $m > 0$, let $D_{\mathbb{Z}^m,\sigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter $\sigma > 0$. We use the following lemmas regarding distributions.

Lemma 14 ([Reg09], Lemma 2.5)**.** *We have* $\Pr[\|\mathbf{x}\|_2 > \sigma\sqrt{m} : \mathbf{x} \xleftarrow{\$} D_{\mathbb{Z}^m,\sigma}] < 2^{-2m}$.

Lemma 15 (Leftover Hash Lemma)**.** *Let $q > 2$ be a prime, m, n, k be positive integers such that $m > (n+1)\log q + \omega(\log n)$, $k = \mathsf{poly}(n)$. Then, if we sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} \xleftarrow{\$} \{-1,0,1\}^{m\times k}$, then $(\mathbf{A}, \mathbf{AR})$ is distributed negligibly close to $U(\mathbb{Z}_q^{n\times m}) \times U(\mathbb{Z}_q^{n\times k})$.*

Gadget Matrix. Let $n, q \in \mathbb{Z}$ and $m \geq n\lceil\log q\rceil$. A gadget matrix $\mathbf{G}$ is defined as $\mathbf{I}_n \otimes (1, 2, ..., 2^{\lceil\log q\rceil-1})$ padded with $m - n\lceil\log q\rceil$ zero columns. For any t, there exists an efficient deterministic algorithm $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times t} \to \{0,1\}^{m\times t}$ that takes $\mathbf{U} \in \mathbb{Z}_q^{n\times t}$ as input and outputs $\mathbf{V} \in \{0,1\}^{m\times t}$ such that $\mathbf{GV} = \mathbf{U}$.

Trapdoors. We summarize properties of lattice trapdoors based on the presentation by Brakerski and Vaikuntanathan [BV16]. Let $n, m, q \in \mathbb{N}$ and consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. For all $\mathbf{V} \in$

[15] This assumption is also called search bilinear Diffie-Hellman assumption.

$\mathbb{Z}_q^{n\times m'}$, we let $\mathbf{A}_\sigma^{-1}(\mathbf{V})$ be a distribution that is a Gaussian $(D_{\mathbb{Z}^m,\sigma})^{m'}$ conditioned on $\mathbf{A}\cdot\mathbf{A}_\sigma^{-1}(\mathbf{V}) = \mathbf{V}$. A σ-trapdoor for $\mathbf{A}$ is a procedure that can sample from the distribution $\mathbf{A}_\sigma^{-1}(\mathbf{V})$ in time $\mathsf{poly}(n, m, m', \log q)$ for any $\mathbf{V}$. We slightly overload notation and denote a σ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\sigma^{-1}$. We have the following:

Theorem 17 (Properties of Trapdoors [Ajt96, GPV08, ABB10a, CHKP12, ABB10b, MP12, BLP+13])**.** *Lattice trapdoors exhibit the following properties.*

1. *Given $\mathbf{A}_\sigma^{-1}$, one can obtain $\mathbf{A}_{\sigma'}^{-1}$ for any $\sigma' \geq \sigma$.*
2. *Given $\mathbf{A}_\sigma^{-1}$, one can obtain $[\mathbf{A}\|\mathbf{B}]_\sigma^{-1}$ for any $\mathbf{B}$.*
3. *For all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{R} \in \mathbb{Z}^{m\times N}$ with $N > n\lceil\log q\rceil$, and invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$, one can obtain $[\mathbf{A}\|\mathbf{A}\mathbf{R} + \mathbf{H}\cdot\mathbf{G}]_\sigma^{-1}$ for $\sigma = m\cdot\|\mathbf{R}\|_\infty\cdot\omega(\sqrt{\log m})$.*
4. *There exists an efficient procedure $\mathsf{TrapGen}(1^n, 1^m, q)$ that outputs $(\mathbf{A}, \mathbf{A}_{\sigma_0}^{-1})$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for some $m = O(n\log q)$ and is 2^{-n}-close to uniform, where $\sigma_0 = \omega(\sqrt{n\log q\log n})$.*
5. *For $\mathbf{A}_\sigma^{-1}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, it follows $\Pr[\|\mathbf{A}_\sigma^{-1}(\mathbf{u})\|_\infty > \sqrt{m}\sigma] = \mathsf{negl}(\lambda)$.*

Lemma 16 (Noise Rerandomization)**.** *Let q, m, k be positive integers and r a positive real satisfying $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log k})\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{x} \in \mathbb{Z}_q^m$ chosen from $D_{\mathbb{Z}^m,r}$. Then, there exists a PPT algorithm $\mathsf{ReRand}(\mathbf{V}, \mathbf{b}+\mathbf{x}, r, \sigma)$ that for any $\mathbf{V} \in \mathbb{Z}_q^{m\times k}$ and positive real $\sigma > s_1(\mathbf{V})$, outputs $\mathbf{b}' = \mathbf{b}\mathbf{V} + \mathbf{x}' \in \mathbb{Z}_q^k$ where $\mathbf{x}'$ is distributed statistically close to $D_{\mathbb{Z}^k,2r\sigma}$*

We recall the *full-rank difference* encoding [ABB10a].

Definition 18 (Full-Rank Difference)**.** *Let k, q be integers such that q a prime. A function $\mathsf{H}_k^{\mathsf{frd}} : \mathbb{Z}_q^k \to \mathbb{Z}_q^{k\times k}$ is a* full-rank difference *encoding if the following holds:*

- *For all distinct $\mathbf{x}, \mathbf{x}' \in \mathbb{Z}_q^k$, the matrix $\mathsf{H}_k^{\mathsf{frd}}(\mathbf{x}) - \mathsf{H}_k^{\mathsf{frd}}(\mathbf{x}') \in \mathbb{Z}_q^{k\times k}$ is full rank over modulo q.*
- *$\mathsf{H}_k^{\mathsf{frd}}$ is computable in time $\mathsf{poly}(k, \log q)$.*

Let $g(X)$ be an arbitrary irreducible polynomial in $\mathbb{Z}_q[X]$ of degree $k-1$. Then, for a vector $\mathbf{x} \in \mathbb{Z}_q^k$, we define $\phi : \mathbb{Z}_q^k \to \mathbb{Z}_q[X]/g(X)$ as the polynomial embedding of $\mathbf{x}$, i.e., $\phi(\mathbf{x}) = \sum_{i\in[k]} \mathbf{x}_i X^{i-1} \in \mathbb{Z}_q[X]/g(X)$, where $\mathbf{x}_i$ is the i-th entry of $\mathbf{x}$. We define the inverse operation as $[\cdot]_{\mathsf{coeff}} : \mathbb{Z}_q[X]/g(X) \to \mathbb{Z}_q^k$. It is shown in [ABB10a] that the following function is a full-rank difference encoding.

Lemma 17. *Let k, q be integers such that q a prime. Define the function $\mathsf{H}_k^{\mathsf{frd}} : \mathbb{Z}_q^k \to \mathbb{Z}_q^{k\times k}$ as*

$$\mathsf{H}_k^{\mathsf{frd}}(\mathbf{x}) = \begin{bmatrix} [\phi(\mathbf{x})]_{\mathsf{coeff}} \\ [X\cdot\phi(\mathbf{x}) \mod g(X)]_{\mathsf{coeff}} \\ \vdots \\ [X^{k-1}\cdot\phi(\mathbf{x}) \mod g(X)]_{\mathsf{coeff}} \end{bmatrix}.$$

Then $\mathsf{H}_k^{\mathsf{frd}}$ is a full-rank difference encoding.

We sometimes consider a function $\mathsf{H}^{\mathsf{frd}}$ defined over $\cup_{k\in\mathbb{N}}\mathbb{Z}_q^k$ that takes $\mathbf{x} \in \mathbb{Z}_q^k$ for some k and outputs $\mathsf{H}_k^{\mathsf{frd}}(\mathbf{x})$.

Hardness Assumption. Finally, for our lattice-based constructions, we rely on the learning with errors assumption.

Definition 19 ([Reg09], Learning with Errors)**.** *For integers n, m, a prime $q > 2$, an error distribution χ over $\mathbb{Z}$, and a PPT algorithm $\mathcal{A}$, the advantage for the* learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ *of $\mathcal{A}$ is defined as follows:*

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

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

The hardness of (decisional) $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\sigma}}$ for $\sigma > 2\sqrt{n}$ has been shown by Regev [Reg09] under the worst case hardness of lattice problems.

B.2 Partitioning-Based Reduction for ABB IBE

Here, we provide description of d-extended ABB IBE scheme. The construction is parameterized by a d-wise linearly independent hash function $h_{d\text{-wise}} : \{0,1\}^\ell \to \{0,1\}^{L_d}$ defined in Sec. 5.5. When we set $d = 3$, we recover ABB IBE scheme, where $h_{d\text{-wise}}(\mathsf{ID}) = (1, \mathsf{ID})$ for $d = 3$.[16] In the following, for notational simplicity, we fix d to be some value and denote L_d and $h_{d\text{-wise}}$ as L and h, respectively. We provide the description of d-extended ABB in Fig. 4.

$\mathsf{Setup}(1^\lambda)$

1 : $(\mathbf{A}, \mathbf{A}_{\sigma_0}^{-1}) \xleftarrow{\$} \mathsf{TrapGen}(1^n, 1^m, q)$
2 : **for** $i \in [L]$ **do** $\mathbf{B}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$
3 : $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$
4 : $\mathsf{mpk} := (\mathbf{A}, (\mathbf{B}_i)_{i \in [L]}, \mathbf{u})$
5 : $\mathsf{msk} := \mathbf{A}_{\sigma_0}^{-1}$
6 : **return** $(\mathsf{mpk}, \mathsf{msk})$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$

1 : $\mathbf{B}_{\mathsf{ID}} := \sum_{i : h(\mathsf{ID})_i = 1} \mathbf{B}_i$
2 : $\mathbf{e}_{\mathsf{ID}} \xleftarrow{\$} [\mathbf{A} \| \mathbf{B}_{\mathsf{ID}}]_\sigma^{-1}(\mathbf{u})$
3 : $\mathsf{sk}_{\mathsf{ID}} := \mathbf{e}_{\mathsf{ID}}$
4 : **return** $\mathsf{sk}_{\mathsf{ID}}$

$\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$

1 : $\mathbf{B}_{\mathsf{ID}} := \sum_{i : h(\mathsf{ID})_i = 1} \mathbf{B}_i$
2 : $(\mathbf{s}, \mathbf{z}_1, z') \xleftarrow{\$} D_{\mathbb{Z}^n, \sigma_1} \times D_{\mathbb{Z}^m, \sigma_1} \times D_{\mathbb{Z}, \sigma_1}$
3 : $\mathbf{z}_2 \xleftarrow{\$} D_{\mathbb{Z}^m, \sigma_2}$
4 : $\mathsf{ct}^{(1)} := \mathbf{s}\mathbf{A} + \mathbf{z}_1$
5 : $\mathsf{ct}^{(2)} := \mathbf{s}\mathbf{B}_{\mathsf{ID}} + \mathbf{z}_2$
6 : $\mathsf{ct}^{(3)} := \mathbf{s}\mathbf{u}^\top + z' + \lfloor q/2 \rceil \cdot \mathsf{M}$
7 : $\mathsf{ct} \leftarrow (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)})$
8 : **return** ct

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct})$

1 : $w := \mathsf{ct}^{(3)} - [\mathsf{ct}^{(1)} \| \mathsf{ct}^{(2)}] \mathbf{e}_{\mathsf{ID}}^\top$
2 : **if** $|w| < q/4$ **do**
3 : **return** 0
4 : **return** 1

Figure 4: ABB IBE Scheme.

[16] In fact, it is a slight variant of the original scheme provided in [ABB10a], where the encryption algorithm is simplified using the proof technique by Katsumata and Yamada [KY16]. Our technique is agnostic to this modification.

The following Lemma 18 establishes the existence of a partitioning-based reduction for the d-extended ABB IBE scheme shown in Sec. 5.5.

Lemma 18. *For any d, there is a $(\mathsf{negl}(\lambda), 0, \mathsf{negl}(\lambda), 0)$-partitioning-based reduction for the d-extended ABB IBE scheme from the* LWE *problem with respect to the partitioning function with approximation $\mathsf{F}_{\mathsf{ParWat}}$ in Sec. 5.5. Concretely, we can choose the following asymptotic parameters for the scheme:*

- $\sigma_0 = \omega(\sqrt{n \log q \log n})$. *(For* TrapGen *in Theorem 17.)*
- $m > (n+1)\log q + \omega(\log n)$. *(For left over hash lemma, Lemma 15.)*
- $\sigma = mL \cdot \omega(\sqrt{\log m})$. *(For sampling $\mathsf{sk}_{\mathsf{ID}}$ with* SimKeyGen*.)*
- $\sigma_1 = 2\sqrt{n}$. *(For* LWE *problem to be difficult.)*
- $\sigma_2 = \sqrt{n}mL \cdot \omega(\sqrt{\log m})$. *(For noise rerandomization lemma, Lemma 16.)*
- $q = nm^2 L \cdot \omega(\log m)$. *(For correctness.)*

Proof. We define the simulation algorithms in Fig. 5. It is easy to check that the running time of each algorithms are $\mathsf{poly}(\lambda)$ related to their counterpart real algorithms. Below, we check all the properties Def. 11 required by a partitioning-based reduction.

Master public key simulatability. For $\psi \in \mathcal{D}_0 \cup \mathcal{D}_1$, $(\mathbf{A}, \mathbf{u})$ are uniformly random over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$. Then, due to Lemma 15, $(\mathbf{A}, \mathbf{A}[\mathbf{R}_1\| \cdots \|\mathbf{R}_L])$ is distributed negligibly close to uniform over $\mathbb{Z}_q^{n\times m} \times (\mathbb{Z}_q^{n\times m})^L$. This implies that mpk output by SimSetup and Setup are distributed negligibly close.

Secret key simulatability. Let us fix any $\mathsf{mpk} = (\mathbf{A}, (\mathbf{B}_i)_{i\in[L]}, \mathbf{u})$ with a corresponding $\mathsf{msk} = \mathbf{A}_{\sigma_0}^{-1}$. Any $\mathsf{sk}_{\mathsf{ID}} = \mathbf{e}_{\mathsf{ID}}$ sampled from $[\mathbf{A}\|\mathbf{B}_{\mathsf{ID}}]_\sigma^{-1}(\mathbf{u})$ is distributed as a discrete Gaussian $D_{\mathbb{Z}^{2m},\sigma}$ conditioned on $[\mathbf{A}\|\mathbf{B}_{\mathsf{ID}}]\mathbf{e}_{\mathsf{ID}} = \mathbf{u}$. We argue that this is the same distribution as a sample from $[\mathbf{A}\|\mathbf{A}\mathbf{R}_{\mathsf{ID}} + \mathbf{H}_{\mathsf{ID}} \cdot \mathbf{G}]_\sigma^{-1}(\mathbf{u})$. First,

$$\begin{aligned}\mathbf{B}_{\mathsf{ID}} &= \sum_{i:h(\mathsf{ID})_i=1} \mathbf{B}_i \\ &= \sum_{i:h(\mathsf{ID})_i=1} \left(\mathbf{A}\mathbf{R}_i + \mathsf{H}_n^{\mathsf{frd}}(K_i)\mathbf{G}\right) = \mathbf{A}\mathbf{R}_{\mathsf{ID}} + \mathbf{H}_{\mathsf{ID}} \cdot \mathbf{G}.\end{aligned}$$

Next, by the assumption that $\mathsf{F}_{\mathsf{ParWat}}(K, \mathsf{ID}) = 1$, we have $\mathbf{H}_{\mathsf{ID}} \neq \mathbf{0}_{n\times n}$. Moreover, due to our specific choice of full-rank difference encoding (see Lemma 17), $\mathbf{H}_{\mathsf{ID}}$ is full rank over q, and hence, invertible since we assume q a prime. We also have $\|\mathbf{R}_{\mathsf{ID}}\|_\infty \leq L$ since each $\mathbf{R}_i \in \{-1, 0, 1\}$ for $i \in [L]$. Then, due to our parameter selection and Theorem 17, $[\mathbf{A}\|\mathbf{A}\mathbf{R}_{\mathsf{ID}} + \mathbf{H}_{\mathsf{ID}} \cdot \mathbf{G}]_\sigma^{-1}(\mathbf{u})$ indeed produces the same distribution as $[\mathbf{A}\|\mathbf{B}_{\mathsf{ID}}]_\sigma^{-1}(\mathbf{u})$.

Ciphertext simulatability. Let us fix any $\mathsf{mpk} = (\mathbf{A}, (\mathbf{B}_i)_{i\in[L]}, \mathbf{u})$ with a corresponding $\mathsf{td} = \big((\mathbf{R}_i)_{i\in[L]}, \psi\big)$, where $\psi = (\mathbf{A}, \mathbf{u}, \mathbf{b}, b')$ with $\mathbf{b} = \mathbf{s}\mathbf{A} + \mathbf{z}$ and $b' = \mathbf{s}\mathbf{u}^\top + z'$, that is, $\psi \in \mathcal{D}_0$. Notice that $\mathsf{ct}^{(1)} = \mathbf{b}$ and $\mathsf{ct}^{(3)} = b'$ are distributed identically for both Encrypt and SimEncrypt. We thus focus on $\mathsf{ct}^{(2)}$. Following the above argument, we have

$$\mathbf{s}\mathbf{B}_{\mathsf{ID}} + \mathbf{z}_2 = \mathbf{s}(\mathbf{A}\mathbf{R}_{\mathsf{ID}} + \mathbf{H}_{\mathsf{ID}} \cdot \mathbf{G}) + \mathbf{z}_2 = \mathbf{s}\mathbf{A}\mathbf{R}_{\mathsf{ID}} + \mathbf{z}_2$$

for $\mathbf{z}_2 \xleftarrow{\$} D_{\mathbb{Z}^m,\sigma_2}$, where the third equality follows from the assumption that $\mathsf{F}_{\mathsf{ParWat}}(K,\mathsf{ID}) = 0$. Here, note that $s_1(\mathbf{R}_{\mathsf{ID}}) \leq m\ell$. On the other hand, when $\mathbf{b} = \mathbf{sA} + \mathbf{z}$ with $\mathbf{z} \xleftarrow{\$} D_{\mathbb{Z}^m,\sigma_1}$, due to the noise rerandomization lemma (see Lemma 16) and our parameter selection, we have $\mathbf{b}_{\mathsf{ID}} = \mathbf{sAR}_{\mathsf{ID}} + \mathbf{z}'_2$, where $\mathbf{z}'_2$ is distributed negligible close to $D_{\mathbb{Z}^m,\sigma_2}$. Hence, the distribution of $\mathsf{ct}^{(2)}$ in $\mathsf{Encrypt}$ and $\mathsf{SimEncrypt}$ are negligibly close as desired.

Ciphertext randomizability. Observe that in case $\psi \xleftarrow{\$} \mathcal{D}_1$, then $(\mathbf{b}, b')$ are uniformly random over $\mathbb{Z}_q^m \times \mathbb{Z}_q$ and independent from mpk. Therefore, $\mathsf{ct}^{(3)}$ is distributed uniformly random for all $\mathsf{M} \in \mathcal{M}$. Moreover, $\mathsf{ct}^{(1)}$ and $\mathsf{ct}^{(2)}$ are distributed independently of M. This completes the proof. □

$\mathsf{SimSetup}(K, \psi)$

1 : **parse** $(K_i)_{i\in[L]} \leftarrow K$
2 : **parse** $(\mathbf{A}, \mathbf{u}, \mathbf{b}, b') \leftarrow \psi$
3 : **for** $i \in [L]$ **do**
4 : $\quad \mathbf{R}_i \xleftarrow{\$} \{-1, 0, 1\}^{m\times m}$
5 : $\quad \mathbf{B}_i := \mathbf{AR}_i + \mathsf{H}_n^{\mathsf{frd}}(K_i)\mathbf{G}$
6 : $\mathsf{mpk} := (\mathbf{A}, (\mathbf{B}_i)_{i\in[L]}, \mathbf{u})$
7 : $\mathsf{td} := \big((\mathbf{R}_i)_{i\in[L]}, \psi\big)$
8 : **return** $(\mathsf{mpk}, \mathsf{td})$

$\mathsf{SimEncrypt}(\mathsf{td}, \mathsf{ID}, \mathsf{M})$

1 : $\mathbf{R}_{\mathsf{ID}} := \sum_{i:h(\mathsf{ID})_i=1} \mathbf{R}_i$
2 : $\mathbf{b}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{ReRand}(\mathbf{R}_{\mathsf{ID}}, \mathbf{b}, \sigma_1, \frac{\sigma_2}{2\sigma_1})$
3 : $\mathsf{ct}^{(1)} := \mathbf{b}$
4 : $\mathsf{ct}^{(2)} := \mathbf{b}_{\mathsf{ID}}$
5 : $\mathsf{ct}^{(3)} := b' + \lfloor q/2 \rceil \cdot \mathsf{M}$
6 : $\mathsf{ct} \leftarrow (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)})$
7 : **return** ct

$\mathsf{SimKeyGen}(\mathsf{td}, \mathsf{ID})$

1 : $\mathbf{R}_{\mathsf{ID}} := \sum_{i:h(\mathsf{ID})_i=1} \mathbf{R}_i$
2 : $\mathbf{H}_{\mathsf{ID}} := \sum_{i:h(\mathsf{x})_i=1} \mathsf{H}_n^{\mathsf{frd}}(K_i)$
3 : **abort if** $\mathbf{H}_{\mathsf{ID}}$ is non-invertible over $\mathbb{Z}_q$
4 : $\mathbf{e}_{\mathsf{ID}} \xleftarrow{\$} [\mathbf{A} \| \mathbf{AR}_{\mathsf{ID}} + \mathbf{H}_{\mathsf{ID}} \cdot \mathbf{G}]_\sigma^{-1}(\mathbf{u})$
5 : $\mathsf{sk}_{\mathsf{ID}} := \mathbf{e}_{\mathsf{ID}}$
6 : **return** $\mathsf{sk}_{\mathsf{ID}}$

Hard Distribution $\mathcal{D}_0$

1 : $(\mathbf{A}, \mathbf{u}) \xleftarrow{\$} \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$
2 : $(\mathbf{s}, \mathbf{z}, z) \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma_1} \times D_{\mathbb{Z}^m,\sigma_1} \times D_{\mathbb{Z}^n,\sigma_1}$
3 : $\mathbf{b} := \mathbf{sA} + \mathbf{z}$
4 : $b' := \mathbf{su}^\top + z'$
5 : **return** $\psi_0 := (\mathbf{A}, \mathbf{u}, \mathbf{b}, b')$

Hard Distribution $\mathcal{D}_1$

1 : $(\mathbf{A}, \mathbf{u}) \xleftarrow{\$} \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$
2 : $(\mathbf{b}, b') \xleftarrow{\$} \mathbb{Z}_q^m \times \mathbb{Z}_q$
3 : **return** $\psi_1 := (\mathbf{A}, \mathbf{u}, \mathbf{b}, b')$

Figure 5: Algorithms used by the partitioning-based reduction for the ABB IBE scheme.

We note that we can also prove partitioning based reduction for ABB IBE scheme with respect to $\mathsf{F}_{\mathsf{Boy}}$ by the very similar analysis. However, the final reduction cost obtained by the analysis using $\mathsf{F}_{\mathsf{Boy}}$ is worse than that obtained by $\mathsf{F}_{\mathsf{Boy}}$ with $d = 3$. We therefore omit the details.

B.3 Proof of Theorem 10

Theorem 18 (Restate of Theorem 10)**.** *If there is an* $(t_{\mathsf{A}}, Q, \epsilon_{\mathsf{A}})$*-adversary* A *against the* IND-CPA *security of the* d*-extended ABB IBE scheme for odd integer* $d \geq 3$*, there is an adversary* B *that*

breaks the LWE problem with advantage ϵ_{B} *and* t_{B} *such that*

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}^{1+\frac{1}{d-1}}}{12qQ} - \mathsf{negl}(\lambda), \quad t_{\mathsf{B}} = t_{\mathsf{A}} + Q \cdot \mathsf{poly}(\lambda).$$

In particular, if we have $d \geq \omega(1)$*, we have*

$$\epsilon_{\mathsf{B}} > \frac{\epsilon_{\mathsf{A}}}{12q\lambda Q} - \mathsf{negl}(\lambda), \quad t_{\mathsf{B}} = t_{\mathsf{A}} + Q \cdot \mathsf{poly}(\lambda)$$

where $q^n \geq 2 \cdot Q \cdot \epsilon^{-\frac{1}{d-1}}$ *holds for dimension* n *of the scheme and* $\mathsf{poly}(\lambda)$ *is roughly the overhead incurred by the running the simulated algorithms compared to the real* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt})$ *algorithms.*

Proof. The proof can be directly obtained by applying Theorem 4 and Lemma 18 to Theorem 7. □