BERGISCHE
UNIVERSITÄT
WUPPERTAL

Doctoral Dissertation

# More Efficient Techniques for Adaptively-Secure Cryptography

**David Niehues**, M.Sc.

September 28, 2021

Submitted to the
School of Electrical, Information and Media Engineering
University of Wuppertal

for the degree of
Doktor-Ingenieur (Dr.-Ing.)

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-urn:nbn:de:hbz:468--20220111-120242-4
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20220111-120242-4]

DOI: 10.25926/rdtq-jw45
[https://doi.org/10.25926/rdtq-jw45]

**David Niehues**
Place of birth: Dortmund, Germany
Author's contact information:
david.niehues@uni-wuppertal.de

| | |
|---|---|
| Thesis Advisor: | **Prof. Dr.-Ing. Tibor Jager**<br>University of Wuppertal, Wuppertal, Germany |
| Second Examiner: | **Prof. Dr. rer. nat. Anja Lehmann**<br>Hasso Plattner Institute, Potsdam, Germany |
| Thesis submitted: | September 28, 2021 |
| Thesis defense: | December 03, 2021 |
| Last revision: | December 10, 2021 |

# Acknowledgements

First, I want to thank Tibor Jager for giving me the opportunity to pursue this research and teaching me to approach cryptography in an intuitive way. I also want to thank him for creating a supportive and inclusive working environment.

Moreover, I want to thank my colleagues at the chair of IT-Security and Cryptography for many insightful discussions and the time spent together: Saqib, Pascal, Kai, Peter, Rafael, Gareth, Lin, Denis, Jan, Tobias, and Jutta. In particular, I want to thank my co-author Rafael for the fruitful discussions and for staying up late to finish our joined work when necessary. Furthermore, I want to thank the student workers Timo, Sven, Tom, and Dennis for showing me what can be achieved together. It was a pleasure working with you. I also want to thank the CRC 901 for supporting me and providing me with the opportunity to experience what research is like early on during my studies and showing me the potential of interdisciplinary research.[1]

Last but not least, I want to thank my family and friends, without who I could not have achieved this. Most of all, I want to thank my parents Irmgard and Helmut, for supporting me throughout my studies and believing in me. I want to thank Kira for making me understand how long of a way confidence can go. I want to thank Benedikt for being a great friend for as long as I can think and helping me to keep my feet on the ground. I want to thank Rike for reminding me that it is worth pursuing one's dreams. Sebi and Mark, I want to thank you for making me realize the value of pragmatism. I want to thank Marcel and Thorsten for accompanying me through my studies. Finally, I also want to thank everyone that I could not give the credit they deserve.

[1]This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre On-The-Fly Computing (GZ: SFB 901/3) under the project number 160364472.

# Contents

# Acronyms

**AHF** admissible hash function

**bAHF** balanced admissible hash function

**BCH** Bose–Chaudhuri–Hocquenghem

**cAHF** computational admissible hash function

**DEM** data encapsulation mechanism

**ECC** error-correcting code

**ECR** exponentially-collision resistant

**ELF** extremely lossy function

**eSIS** exponential short integer solution

**ETH** Exponential Time Hypothesis

**FRD** full rank difference encoding function

**GV** Gilbert-Varshamov

**IB-KEM** identity-based key encapsulation mechanism

**IBE** identity-based encryption

**LWE** learning with errors

**MRRW** McEliece-Rodemich-Rumsey-Welch

**NIZK** non-interactive zero-knowledge proof

**PHF** programmable hash function

**PPT** probabilistic polynomial time algorithm

**PRF** pseudorandom function

**RO** random oracle

**ROM** random oracle model

**SIS** short integer solution

**TCR** truncation-collision resistant

**TM** Turing machine

**VRF** verifiable random function

**VUF** verifiable unpredictable function

**wNCR** weak near-collision resistant

# 1 Introduction

Modern-day communication has become more and more digital. While this has made human life more convenient and many processes in society and economy more efficient, it has also created more and more opportunities for various adversaries to manipulate communication or eavesdrop on it. The Snowden revelations in 2013 further highlighted the seriousness of these threats. To protect the communication of people, companies, and states from such threats, we require cryptography. Throughout the introduction, we first give a brief overview of the principles of cryptography as a science and then describe the contributions of this thesis to the field of cryptography.

## 1.1 Modern Cryptography

Ronald Rivest very fittingly described cryptography as being "[...] about the communication in the presence of adversaries" [Riv90, p. 719]. Taking encryption as an example, we naturally require that an encryption scheme guarantees the *confidentiality* of the communication in the presence of adversaries. However, this intuitive definition of security for encryption schemes leaves some questions unanswered. For example, the capabilities of potential adversaries may differ significantly depending on the application in which an encryption scheme is used. It may be realistic to assume that all potential adversaries can only passively eavesdrop on exchanged messages in some applications. Meanwhile, this assumption may not hold for other applications where adversaries can trick communication partners into encrypting specific messages of the adversary's choice. Moreover, even if encrypting the messages guarantees confidentiality of the communication, it might not prevent adversaries from manipulating the content of the messages without learning anything about the content.

These open questions highlight the need for formal definitions of security. Thus, modern cryptography requires mathematically precise definitions of security for cryptographic schemes. Furthermore, there can be several equally reasonable definitions of security that are fitting in different applications. For example, some applications may have the property that all encrypted messages are distributed uniformly at random. Moreover, as discussed above, we have to consider adversaries with different capabilities for different applications. We formalize these security requirements in modern cryptography in so-called *security experiments* or sometimes also called *security games*. Such a security experiment is modeled as an interaction between two algorithms: a *challenger* and an *adversary*.

Provable security. For a long time, cryptographic schemes were proposed and considered secure if no successful attack was found for an extended period of time. However, this approach to security left open the possibility of arbitrary new attacks that could compromise the security of a cryptographic scheme. Goldwasser and Micali addressed this issue in 1984 by introducing the paradigm of *provable security* [GM84]. Following this paradigm, we consider an encryption scheme secure if there is a mathematical sound proof showing that any algorithm that can break the security of a scheme would imply an algorithm that can efficiently solve a computational problem that is believed to be intractable. More specifically, we consider a computational problem that is believed to be impossible to solve with probability $\varepsilon$ by any algorithm that runs in time at most $t$. We then show that any algorithm $\mathcal{A}$ that runs in time $t_{\mathcal{A}} \in \mathbb{N}$ and can break the security of a cryptographic scheme with probability $\varepsilon_{\mathcal{A}}$ implies the existence of an algorithm $\mathcal{B}$ that can solve the computational problem that is believed to be intractable

$$\text{in time} \qquad t_{\mathcal{B}} \leq t \qquad \text{with probability} \qquad \varepsilon_{\mathcal{B}} \geq \varepsilon.$$

Thus, the existence of an algorithm that can break the security of the cryptographic scheme contradicts the assumed intractability of the computational problem. Examples of such problems used in cryptography are the discrete logarithm problem, the RSA (Rivest-Shamir-Adleman) problem, and the short integer solution problem. These computational problems have been extensively studied, and so far, no practically efficient algorithm has been found for cryptographically relevant parameter sizes.

Kerckhoffs' principle. Another essential paradigm in cryptography that is orthogonal to the paradigms we discussed above is *Kerckhoffs' principle*. Translated to English, it states that any cryptographic "system must not require secrecy and can be stolen by the enemy without causing trouble" [Pet11, p. 675]. That is, for example, an encryption scheme should remain secure as long the decryption key is kept secret, even if all details of the scheme become known to the adversary. Thus, the security of the scheme should follow only from the secrecy of the key. The principle was first described by Auguste Kerckhoff as one of six principles [Ker83a, Ker83b].

Following this principle has several practical advantages. First and foremost, the security of a cryptographic scheme that is publicly available can be validated by a large number of experts. Moreover, it is practically much easier to keep a small key secret than a complete encryption scheme, and if a secret key becomes public, it is much easier to replace. For these reasons, the vast majority of cryptographic algorithms used in everyday life are publicly known.

## 1.2 Overview of this Thesis

We give a brief overview of the topics covered in this thesis. First, we informally describe the cryptographic primitives this thesis focuses on, namely *identity-based encryption* (IBE)[1] and *verifiable random function*s (VRFs). We then informally discuss *adaptive security*, the type of security model this thesis is concerned with before presenting the outline of the thesis.

### 1.2.1 Richer forms Cryptography

While cryptography is associated the most with encryption, this is only one of many primitives modern cryptography research is concerned with. This thesis presents novel *identity-based encryption* (IBE) schemes and *verifiable random function*s (VRFs). Therefore, we informally introduce these two primitives below.

Identity-based encryption. *Identity-based encryption* (IBE) is a variant of encryption that Shamir first proposed in 1984 [Sha84]. In an IBE scheme, a trusted third party provides publicly known system parameters and personal secret keys to all users. After this setup step, the knowledge of a receiver's identity, *e.g.*, their e-mail address, suffices to encrypt a message such that only the receiver can correctly decrypt the ciphertext. Thus, this form of encryption makes the distribution of keys among users obsolete and only requires the distribution of the system parameters. Even though Shamir already proposed the concept in 1984, it was only in 2001 when Boneh and Franklin [BF01] and Cocks [Coc01] independently presented the first IBE schemes with practical efficiency.

Verifiable random functions. The second cryptographic primitive this thesis focuses on are *verifiable random function*s (VRFs), which are an extension to *pseudorandom function*s (PRFs). Therefore, we briefly describe PRFs as a prerequisite to describe VRFs.

Given $m, n \in \mathbb{N}$ with $m \leq n$, consider the set $\mathcal{F} = \{f : \{0,1\}^m \to \{0,1\}^n\}$, *i.e.* the set of all functions that map bit strings of length $m$ to bit strings of length $n$. Intuitively, a function $f$ drawn uniformly at random from $\mathcal{F}$ is helpful in cryptography since evaluating $f$ on an arbitrary input yields an independent uniformly random output, thus perfectly hiding the input. However, since there are $2^{n(2^m)}$ different functions in $\mathcal{F}$, it would take at least $n \cdot 2^m$ bits to describe a function from $\mathcal{F}$ uniquely. For $m$ and $n$ in the order of 128, which would be a plausible size for cryptography, the size of the function description is too large by many orders of magnitude. Thus, Goldreich *et al.* instead considered a small subset $\mathcal{F}' \subsetneq \mathcal{F}$, where each function $f_K \in \mathcal{F}'$ is efficiently described by a small

[1] This thesis considers *identity-based key encapsulation mechanism*s instead of IBEs. Here, we describe IBEs, because this primitive is better suited for an informal presentation. However, the two primitives are very closely related, and we explain their relationship in Section 2.3.2.

key $K$ [GGM84, GGM86]. Furthermore, Goldreich *et al.* require that for $f$ drawn uniformly at random from $\mathcal{F}$ and $f'$ drawn uniformly at random from $\mathcal{F}'$, there is no practically efficient algorithm that can distinguish $f$ from $f'$ only from the function outputs. We refer to $\mathcal{F}'$ as a family of *pseudorandom function*s (PRFs) if these properties are achieved. We note that with the knowledge of $K$, this indistinguishability does not hold anymore, and therefore $K$ must be kept secret.

Pseudorandom functions have the downside that an entity evaluating the PRF for a user can lie to the user about the outputs of the PRF without the user being able to take notice. *Verifiable random function*s (VRFs), introduced by Micali, Rabin, and Vadhan in 1999 [MRV99], address this issue by making the output of a PRF *verifiable*. That is, a secret key $\mathsf{sk}$ identifies a function $F_{\mathsf{sk}}$, but in contrast to PRFs, the private key always comes together with a public verification key $\mathsf{vk}$. The secret key $\mathsf{sk}$ then allows the evaluation of $F_{\mathsf{sk}}(X)$ on input $X$ and to obtain the pseudorandom output $Y$. However, a VRF also produces a non-interactive proof of correctness $\pi$. Together with $\mathsf{vk}$, the proof $\pi$ allows everyone to verify that $Y$ is $F_{\mathsf{sk}}(X)$. We require two security properties from VRFs: *unique provability* and *pseudorandomness*. Unique provability means that for every verification key $\mathsf{vk}$ and every VRF input $X$, there is a *unique* $Y$ for which a proof $\pi$ exists such that the verification algorithm accepts. However, note that there might be multiple valid proofs $\pi$ verifying the correctness of $Y$ with respect to $\mathsf{vk}$ and $X$. Furthermore, we (informally) say that a VRF is *pseudorandom* if there is no efficient adversary that can distinguish a VRF output without the accompanying proof from a uniformly random element of the range of the VRF.

### 1.2.2 Adaptive Security

As discussed above in Section 1.1, different applications may require different security properties from cryptographic schemes. This thesis concentrates on security requirements that prevent attacks from *adaptive adversaries*. For example, adaptive adversaries against encryption schemes are characterized by their ability to make communication partners encrypt arbitrary messages of the adversary's choice. Moreover, the adversaries we consider may choose the messages they trick the communication partners to encrypt depending on the ciphertexts the adversary has seen previously. We refer to schemes that are secure in the presence of such adversaries as *adaptively-secure*. Moreover, note that similar adaptive security requirements also apply to IBEs schemes and VRFs, and we discuss them in detail in Section 2.3.

Plausibility of adaptive attacks. Considering adversaries that can obtain the encryption of arbitrary messages of their choice might look like an unnecessarily strong security requirement. However, there are historical and practical examples of attacks that are well captured by this type of adversary. We follow [KL14, Section 3.4.2] and present two examples for this.

Historically, the intelligence operations of the USA in preparation for the Battle of Midway are an example of such an adaptive attack. They are described as follows in Chapter 31 of [LPC85] by Layton *et al.*, who took part in this intelligence operation. The US Navy knew from intercepted and deciphered communications that the Japanese Navy planned to attack an island the Japanese referred to as `AF` in their communications. The US government was aware of an upcoming attack by the Japanese and suspected that `AF` corresponded to the Midway Islands, but they were not sure. To confirm the suspicion, they send a message through an undersea cable to the military base on Midway Islands and asking them to announce a drinking water shortage on the island via an unencrypted radio message. Only 24 hours after this message, US Navy intelligence intercepted Japanese communications about bringing an additional two weeks drinking water reserve for the occupation of `AF`. This observation confirmed to the US Navy that `AF` referred to Midway Islands.

However, there are also examples more aligned with modern communication, like the following one given by [KL14, Section 3.4.2]. Consider an adversary with remote access to a command-line interface, which encrypts all commands entered by the adversary. If the adversary can eavesdrop on the ciphertexts produced by the command line, then the adversary can obtain ciphertexts for arbitrary messages of its choice.

### 1.2.3 The Random Oracle Model

The seminal work by Goldwasser and Micali introducing the provable security paradigm for cryptography [GM84] was a major step forward for cryptography as a science. However, back then and even today, most cryptographic schemes used in practice are not shown to be secure purely in the models of provable security. Bellare and Rogaway, therefore, introduced the *random oracle model* (ROM) to bridge this gap in their seminal work in 1993 [BR93]. In the ROM, the challenger and the adversary are given oracle access to a shared *random oracle* (RO). If the RO is queried on any input $X$, it checks whether it has been queried for $X$ previously or not. If it has not been queried on $X$ before, it draws an output $Y_X$ uniformly at random from the range of the RO, stores it for future reference. It then returns $Y_X$ to the algorithm that queried for $X$. If the RO has been queried for $X$ previously, it retrieves $Y_X$ from storage and returns $Y_X$ to the algorithm that queried the RO for $X$. In most applications, the domain of a RO will be $\{0,1\}^*$ and the range will be $\{0,1\}^n$ for some $n \in \mathbb{N}$. However, other domains and ranges are used as well. Once a cryptographic scheme is proven secure using the ROM, the heuristic is to instantiate the RO with a cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^n$. Of course, if the RO has a different domain or range, this must be addressed in the instantiation. Furthermore, numerous security proofs in the ROM, particularly for schemes used in practice, also use the so-called *programmability* of random oracles. That is, the reduction in a security proof adaptively chooses specific outputs of the RO for inputs of the reduction's choice while maintaining the output distribution of the RO.

The random oracle heuristic has proven immensely valuable in constructing cryptographic schemes with practical efficiency whose security holds in real-world applications. However, as Canetti *et al.* showed [CGH98, CGH04], ROs can unfortunately not be instantiated in a theoretically sound way. From a theoretical point of view, it would therefore be preferable to have cryptographic constructions that are secure without relying on an uninstantiable model. We say that a construction or proof is in the *standard model* if it does not make use of the ROM. This thesis makes progress towards efficient constructions without the ROM by proposing assumptions for hash functions that are significantly weaker than random oracles and by showing how these assumptions can be used for efficient cryptographic schemes with provable security.

### 1.2.4 Outline

We conclude the introduction by providing an outline of the thesis.

**Chapter 2: Preliminaries** We provide preliminaries to the subsequent chapters of the thesis. That is, we introduce the notation and the computational model used throughout the thesis. Moreover, we formally introduce verifiable random functions and identity-based key encapsulation mechanisms, the cryptographic primitives with which this is concerned the most. Furthermore, we formally introduce the complexity assumptions used in this thesis and describe the game hopping technique to structure complex security proofs. Finally, we discuss *partitioning arguments*, the proof technique that we employ in all constructions of cryptographic schemes in this thesis. Specifically, we first introduce partitioning arguments and discuss how they are used to achieve cryptographic schemes with adaptive security. We then discuss the challenges in using partitioning arguments to prove the adaptive security of schemes that require decisional security, like VRFs and IB-KEMs.

**Chapter 3: Efficient Verifiable Random Functions** This chapter discusses the efficiency of VRFs in the standard model and provides novel constructions that improve on the status quo. To that end, we first introduce *admissible hash function*s (AHFs), a building block used in most recently published VRFs that focus on efficiency and do not rely on the ROM. We then develop tools to assess the potential real-world efficiency of VRFs that rely on AHFs. These tools show that the efficiency that VRFs based on AHFs can achieve is inherently limited by the information-theoretic security required by all instantiations of AHFs known to us. Therefore, we propose *computational admissible hash function*s (cAHFs) as an alternative with computational efficiency, functioning as a drop-in replacement for AHFs. We then present an efficient VRF whose security can be proven using cAHFs. However, these VRFs still have key and proof sizes that are not practical. To construct even more efficient VRFs, we introduce *blockwise partitioning* as a new technique to achieve adaptive security. Blockwise partitioning can no longer function

as a drop-in replacement for AHFs but enables us to construct more efficient VRFs. Finally, we provide a comprehensive comparison of concrete key and proof sizes of VRFs.

**Chapter 4: Verifiable Random Functions with Optimal Tightness** All currently known standard model VRFs have a reduction loss that is much worse than what one would expect from known optimal constructions of closely related primitives like unique signatures. In this chapter, we show that:

1. Every security proof for a VRF that relies on a non-interactive complexity assumption has to lose a factor of $Q$, where $Q$ is the number of adversarial queries. To that end, we extend the meta-reduction technique of Bader *et al.* [BJLS16] to also cover VRFs.
2. This raises the question: Is this bound optimal? We answer this question in the affirmative by presenting the first VRF with a reduction from the non-interactive $q$-DBDHI assumption [BB04a, Definition 4] to the security of VRF that achieves this optimal loss.

We thus paint a complete picture of the achievability of tightly secure verifiable random functions: We show that a security loss of $Q$ is unavoidable and present the first construction that achieves this bound.

**Chapter 5: Efficient Identity-Based Key-Encapsulation Schemes from Lattices** *Programmable hash function*s (PHFs) were formally introduced by Hofheinz and Kiltz in [HK08, HK12], but only for complexity assumptions that do not hold in the presence of large-scale quantum computers. Zhang *et al.* then introduced PHFs for lattices that allow the construction of *identity-based key encapsulation mechanism*s (IB-KEMs), a building block that implies IBEs schemes, and digital signatures with post-quantum security.

In this chapter, we present PHFs with short function descriptions, which thus allow us to significantly reduce the size of public keys and public parameters of digital signatures and IBEs schemes. Moreover, we define *balanced programmable hash function*s (balanced PHFs) that allow the construction of schemes with decisional security, such as IB-KEMs or IBE schemes, without requiring an artificial abort step. Finally, we present an IB-KEM that can be generically be instantiated with balanced PHFs.

## 1.3 Publication Overview

This thesis builds upon the following peer-reviewed publications and the following unpublished manuscript.

**Peer-reviewed publications:**

[JN19a] 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, Waterloo, ON, Canada, August 12–16, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-38471-5_13.

[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, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75245-3_22.

[Nie21] David Niehues. Verifiable random functions with optimal tightness. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 61–91, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75248-4_3.

**Unpublished manuscripts:**

[JN21] Tibor Jager and David Niehues. Compact and Balanced Programmable Hash Functions for Lattices. 2021. Unpublished manuscript. To be submitted to PKC 2022.

# 2 Preliminaries

Before presenting the results of this thesis, we introduce the notation used throughout this thesis and present some further foundations.

## 2.1 Notation

Throughout this thesis, $\lambda$ denotes the security parameter. Such a parameter is commonly used in cryptography and allows to scale the difficulty of breaking a cryptographic scheme. Thus, the security parameter allows to choose key and parameter sizes of cryptographic schemes depending on a desired security level. For $c \in \mathbb{N}$, we denote the set $\{1, \ldots, c\}$ of all integer from 1 to $c$ by by $[c]$. Furthermore, for $a, b \in \mathbb{N}$ with $a \leq b \leq c$ we denote the set $\{a, \ldots, c\}$ by $[a, c]$ and the set $[c] \setminus \{b\}$ by $[c \setminus b]$. Moreover, we let

$$[c]_0 \coloneqq [c] \cup \{0\}, \qquad [a, c]_0 \coloneqq [a, c] \cup \{0\}, \qquad \text{and} \qquad [c \setminus b]_0 \coloneqq [c \setminus b] \cup \{0\}.$$

Also, for a finite set $S$ we denote sampling an element $s$ uniformly at random from $S$ by $s \xleftarrow{\$} S$. Moreover, unless specified otherwise, all logarithms in this thesis are to the basis two and we use $\mathsf{poly}$ to denote an arbitrary polynomial over $\mathbb{R}$ if no further specification is required.

Since we will repeatedly use bit strings and operations on them, we introduce some notation for them. That is, for some $n, i \in \mathbb{N}$ wit $i \leq n$ and a bit string $x \in \{0,1\}^n$, we denote the $i$th bit of $x$ by $x_i$. We will also use this notation for bit strings that are outputs of functions. For example, if $H$ is a function that produces outputs in $\{0,1\}^n$, then we denote the $i$th bit of $H(X)$ by $H(X)_i$, where $X$ is an arbitrary element from $H$'s domain. Furthermore, for $n, m \in \mathbb{N}$ and bit strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ we let

$$x \parallel y \coloneqq x_1, \ldots, x_n, y_1, \ldots, y_m \in \{0,1\}^{n+m}$$

denote the concatenation of $x$ and $y$.

Finally, we introduce the notion of *negligible functions*.

**Definition 1.** We say that a function $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is *negligible* if for every positive polynomial $\mathsf{poly}$ there is an $N \in \mathbb{N}$ such that for all $n \geq N$ it holds that

$$f(n) < \frac{1}{p(n)}.$$

We will denote an arbitrary negligible function by $\mathsf{negl}$. Furthermore, we say that a function $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is *non-negligible* $f$ is not negligible.

## 2.2 Computational Model

We will repeatedly consider runtimes and success probabilities of algorithms and thus require that these quantities are well-defined. We thus define a computational model in the form of *Turing machine*s (TMs), which were first introduced by Alan Turing in [Tur37]. Note that, unless specified otherwise, all algorithms in this thesis are modelled as Turing machines. We follow the notation for Turing machines from [AB09, Chapter 1] and slightly adapt it.

**Definition 2** (Turing machine, adapted from [AB09, Chapter 1])**.** A $k$-tape *Turing machine* (TM) $M$, for $k \in \mathbb{N}$ and $k \geq 2$, is described by a triple $M = (\Sigma, \Gamma, Q, \delta)$ of the following components:

- $\Gamma$ is a finite set of symbols that $M$'s tape can contain. We assume without loss of generality that $\Gamma$ always contains the *blank* symbol $\square$, and the *start symbol* $\triangleright$. We refer to $\Gamma$ as the *tape alphabet* of $M$.
- $\Sigma \subsetneq \Gamma$ is a finite set that contains all symbols that can be an input to $M$. In particular, we require that $\Sigma$ does not contain the start symbol $\triangleright$. We refer to $\Sigma$ as the *input alphabet* of $M$.
- $Q$ is the finite set of states $M$ can be in. We again assume without loss of generality that $Q$ always contains the *start state* $q_{\mathsf{start}}$ and a the *halting state* $q_{\mathsf{halt}}$.
- A *state transition function* $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\mathsf{L}, \mathsf{S}, \mathsf{R}\}^k$. $\delta$ describes the rules that $M$ follows during execution.

If $M$ is in state $q \in Q$ and if $(\sigma_1, \ldots, \sigma_k) \in \Gamma^k$ are the symbols currently being read in the $k$ tapes, and $\delta(q, (\sigma_1, \ldots, \sigma_k)) = (q', (\sigma'_2, \ldots, \sigma'_k), z)$, where $z \in \{\mathsf{L}, \mathsf{S}, \mathsf{R}\}^k$, then at the next step the $\sigma$ symbols in the last $k-1$ tapes will be replaced by the $\sigma'$ symbols, $M$ will be in state $q'$, and the $k$ heads will each move either left, right, or stay, depending on the symbol given in $z$ for the respective tape. If the machine tries to move left from the leftmost position of a tape, then it will stay in place instead.

We refer to the first tape as the *input tape*. Moreover, we refer to triples $(q, X, n) \in Q \times (\Gamma^*)^k \times \mathbb{N}^k$ as a *configurations* of $M$. For such a triple, $q \in Q$ specifies the state $M$ is in, $X = (x_1, \ldots, x_k) \in (\Gamma^*)^k$ specifies the contents of the respective tapes until all further symbols to the right on the respective tape are the blank symbol $\square$. Furthermore, $n = (n_1, \ldots, n_k) \in \mathbb{N}^k$ specifies the positions of $M$'s heads on the respective tapes.

All tapes except for the input tape are initialized with the first symbol being the start symbol $\triangleright$ and with all other locations of the tape being the blank symbol $\square$. The input tape is initialized to begin with the start symbol $\triangleright$, followed by the input of $M$ of finite length, and the blank symbol $\square$ on the rest of its cells. All heads start at the left ends of the tapes and the machine is in the special starting

state $q_{\mathsf{start}}$. We refer to this configuration as the *start configuration* of $M$ on input $x$. Finally, we only allow transition functions $\delta$ that do not allow any change to the state, the content of the tapes or the positions of $M$'s heads once $M$ is in state $q_{\mathsf{halt}}$. The content of the second tape except for trailing blank symbols $\square$ are the output of $M$. For a TM $M$ and $x \in \{0,1\}^*$ we write $M(x)$ to denote the execution of $M$ on input $x$. Furthermore, if there is no specific input $x$, we write use $\cdot$ as a placeholder and write $M(\cdot)$.

Even though Turing machines can use arbitrary input alphabets of finite size, we limit ourselves for to binary inputs for clarity and only consider Turing machines with $\Sigma = \{0,1\}$. Observe that this limitation is without any loss of generality because symbols of arbitrary finite alphabets can be encoded in binary. In particular, we assume that all numbers that are inputs to algorithms are encoded in binary unless stated otherwise. Moreover, we only consider TMs that halt on all inputs.

After formally introducing Turing machines, we proceed by formally defining the running time of Turing machines.

**Definition 3** (Runtime of TMs)**.** We say that a TM $M$ runs in time $t \in \mathbb{N}$ on input $x \in \{0,1\}^*$ if $M$ reaches the state $q_{\mathsf{halt}}$ after exactly $t$ successive evaluations of $\delta$. If there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$ the TM $M$ runs in time at most $p(|x|)$, then way say $M$ runs in polynomial time.

Throughout this thesis, we will repeatedly encounter randomized algorithms that are not captured by the model of computation we described so far. We thus extend our computational model to also capture these algorithms by defining probabilistic Turing machines.

**Definition 4** (Probabilistic TMs, adapted from [Gol08, Definition 6.1])**.** A *probabilistic* Turing machine $M$ is a TM associated with a function $p : \mathbb{N} \to \mathbb{N}$, and with a second designated input tape that takes $p(|x|)$ many uniformly random bits as input, where $x \in \{0,1\}^*$ is $M$'s primary input. For any $x \in \{0,1\}^*$ and $\rho \xleftarrow{\$} \{0,1\}^{p(|x|)}$ we write $M(x;\rho)$ for the execution of $M$ on input $x$ with randomness $\rho$. Moreover, note that the output of $M$ on input $x$ and randomness $\rho \xleftarrow{\$} \{0,1\}^{p(|x|)}$ is a random variable because $\rho$ is drawn uniformly at random. We say that $M$ is a *probabilistic polynomial time algorithm* (PPT) if $p$ is polynomially bounded there is a polynomial $p' : \mathbb{N} \to \mathbb{N}$ such that $M(x;\rho)$ runs in time at most $p'(|x|)$ for all $x \in \{0,1\}^*$ and all $\rho \in \{0,1\}^{p(|x|)}$.

*Remark* 1. In most occasions, we will omit explicitly drawing the randomness $\rho$ and just write $y \xleftarrow{\$} \mathcal{A}(x)$ for executing a probabilistic algorithm $\mathcal{A}$ with input $x$ and uniformly random input $\rho$ of appropriate length. Moreover, for a fixed randomness $\rho$, we view $\mathcal{A}(x;\rho)$ as deterministic algorithm. This will be of relevance in Chapter 4, where we also recall this notation shortly.

In this thesis, we will oftentimes consider algorithms that have oracle access to other algorithms. This interaction is also not captured by our computational model up to this point. We incorporate it by defining *Turing machines with oracle access* as follows.

**Definition 5** (TMs with oracle access, adapted from [Gol08, Definition 1.1])**.** A (probabilistic) Turing machine $M$ with *oracle access* to a (probabilistic) TM $N$ is a Turing machine with a special additional tape, called the *oracle tape*, and two special states, the *oracle invocation state* $q_{\mathsf{invoke}}$ and the *oracle spoke state* $q_{\mathsf{spoke}}$. $M$ accesses its oracle during execution as follows.

- For configurations with a state different from $q_{\mathsf{invoke}}$ the next configuration is defined via the transition function $\delta$ as usual.
- Let $\gamma = (q, X, n)$ be a configuration with $q = q_{\mathsf{invoke}}$ and suppose that the content of the oracle tape (except for trailing blank symbols $\square$) is $o$. Then the configuration following $\gamma$ is identical to $\gamma$, except that the state is $q_{\mathsf{spoke}}$, and the oracle tape contains $N(o)$ (followed by infinitely many blank symbols $\square$). We refer to $o$ as $M$'s query and to $N(o)$ as the oracles response.

We denote the output of $M$ on input $x \in \{0,1\}^*$ with oracle access to $N$ by $M^N(x)$. Moreover, if $N$ is a probabilistic TM, then we implicitly assume that the result is computed as $N(o;\rho)$, where $\rho$ is a bit string of appropriate length that is drawn uniformly at random.

In complexity theory the evaluation of an oracle query is counted as a single step and thus does not affect the asymptotic runtime of a TM with oracle access. However, in some of our applications, it will be convenient to count the evaluation of an oracle query as as many steps as $N$ takes to compute the result. We will explicitly state this for the specific application.

## 2.3 Cryptographic Primitives

As further preliminaries, we now formally introduce VRFs and IB-KEMs, which we informally introduced in Section 1.2.1, and their accompanying security notions.

*Remark* 2. Note that we will view the time to execute the security experiments for VRFs and IB-KEMs as part of the runtime of an adversary that is executed in the security experiment. We do so as to not worsen the runtime of a reduction by accounting it runtime for simulating the security experiment for the adversary. Moreover, it allows us to state the security proofs with more clarity. Moreover, note that we supply the adversary in the security experiments only with $\lambda$ bits of randomness for clarity $\rho_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^\lambda$. Note that this limitation is without loss of generality because the randomness can be stretched to arbitrary polynomial length by a pseudorandom number generator [KL14, Corollary 7.10].

### 2.3.1 Verifiable Random Functions

As we discussed in Section 1.2.1, *verifiable random function*s (VRFs), introduced by Micali, Rabin and Vadhan in [MRV99], can be thought of as the public key

$G^{\mathsf{Psrd}}_{\mathcal{VRF},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$

$(\mathsf{vk},\mathsf{sk}) := \mathsf{SetupVRF}(1^\lambda); \rho_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^\lambda$
$(X^*, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Eval}(\mathsf{sk},\cdot)}(\mathsf{vk}; \rho_{\mathcal{A}})$
$(Y_0, \pi) := \mathsf{Eval}(\mathsf{sk}, X^*)$
$Y_1 \xleftarrow{\$} \mathcal{Y}$
$b \xleftarrow{\$} \{0,1\}$
$b' := \mathcal{A}_2^{\mathsf{Eval}(\mathsf{sk},\cdot)}(Y_b, \mathsf{st})$
**if** $(b' == b)$
  **return** 1
**else**
  **return** 0

*Figure 2.1:* The security experiment specifying pseudorandomness of verifiable random functions.

equivalent of pseudorandom functions. Here we formally introduce VRFs and their security properties.

**Definition 6.** A *verifiable random function* (VRF) with domain $\mathcal{X}$ and finite range $\mathcal{Y}$ consists of three algorithms $\mathcal{VRF} = (\mathsf{SetupVRF}, \mathsf{Eval}, \mathsf{Vfy})$ with the following syntax.

- $(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}(1^\lambda)$ takes as input the security parameter $\lambda$ and outputs a key pair $(\mathsf{vk},\mathsf{sk})$. We say that $\mathsf{sk}$ is the *secret key* and $\mathsf{vk}$ is the *verification key*.
- $(Y,\pi) \xleftarrow{\$} \mathsf{Eval}(\mathsf{sk}, X)$ takes as input a secret key $\mathsf{sk}$ and an input $X \in \mathcal{X}$, and outputs a function value $Y \in \mathcal{Y}$ and a proof $\pi$.
- $\mathsf{Vfy}(\mathsf{vk}, X, Y, \pi) \in \{0,1\}$ takes as input a verification key $\mathsf{vk}$, $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, and proof $\pi$, and outputs a bit.

We say that $\mathcal{VRF} = (\mathsf{SetupVRF}, \mathsf{Eval}, \mathsf{Vfy})$ with domain $\mathcal{X}$ and range $\mathcal{Y}$ is a secure VRF if it fulfills the following requirements.

**Correctness.** For all $(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}(1^\lambda)$, $X \in \mathcal{X}$ and, $(Y,\pi) \xleftarrow{\$} \mathsf{Eval}(\mathsf{sk}, X)$ it must hold that $\mathsf{Vfy}(\mathsf{vk}, X, Y, \pi) = 1$. Further, the algorithms $\mathsf{SetupVRF}$, $\mathsf{Eval}$, $\mathsf{Vfy}$ have to be PPTs.

**Unique provability.** For all $\mathsf{vk} \in \{0,1\}^*$ and all $X \in \mathcal{X}$, there does not *exist* any $Y_0, \pi_0, Y_1, \pi_1 \in \{0,1\}^*$ such that $Y_0 \neq Y_1$, and it holds that $\mathsf{Vfy}(\mathsf{vk}, X, Y_0, \pi_0) = \mathsf{Vfy}(\mathsf{vk}, X, Y_1, \pi_1) = 1$.

$G^{\mathsf{IND\text{-}ID\text{-}CPA}}_{\mathcal{IB}\text{-}\mathcal{KEM},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$

$(\mathsf{mpk}, \mathsf{msk}) \coloneqq \mathsf{SetupIBK}(1^\lambda); \rho_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^\lambda$

$(\mathsf{id}^*, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}; \rho_{\mathcal{A}})$

$\kappa_0 \xleftarrow{\$} \mathcal{K}; (\mathsf{ct}, \kappa_1) \xleftarrow{\$} \mathsf{Encap}(\mathsf{mpk}, \mathsf{id}^*)$

$b \xleftarrow{\$} \{0,1\}$

$b' \coloneqq \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},\cdot)}(\mathsf{ct}, \mathsf{st})$

**if** $(b' == b)$

  **return** 1

**else**

  **return** 0

*Figure 2.2:* The security experiment for *indistinguishability of ciphertexts* under adaptively chosen plaintext attacks for IB-KEMs.

**Pseudorandomness.** Consider an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with access (via oracle queries) to $\mathsf{Eval}(\mathsf{sk}, \cdot)$ in the pseudorandomness game depicted in Figure 2.1. We say that $\mathcal{A}$ is *legitimate* if there is no $\rho_{\mathcal{A}} \in \{0,1\}^\lambda$ such that $\mathcal{A}_1$ or $\mathcal{A}_2$ query $\mathsf{Eval}(\mathsf{sk}, X^*)$, where $X^* \in \mathcal{X}$ is part of the output of $\mathcal{A}_1$. We define the advantage of $\mathcal{A}$ in breaking the pseudorandomness of $\mathcal{VRF}$ as

$$\mathsf{Adv}^{\mathsf{Psrd}}_{\mathcal{VRF},\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[ G^{\mathsf{Psrd}}_{\mathcal{VRF},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1 \right] - 1/2 \right|.$$

In addition to the properties above, Hofheinz and Jager introduced the notion of VRFs with all desired properties [HJ16]. More specifically, a VRF possesses *all desired properties* if it fulfills all requirements above, has an exponentially sized domain and is proven secure under a non-interactive complexity assumption. In this thesis, we only consider VRFs that have all desired properties.

### 2.3.2 Identity-Based Encryption

Identity-based encryption was first described by Shamir in 1984 [Sha84]. As Bentahar *et al.* [BFMS08] describe, IBEs can also be constructed by first constructing an *identity-based key encapsulation mechanism* (IB-KEM) and then combining it with an *data encapsulation mechanism* (DEM). This approach is also known as the KEM-DEM approach and was first formally describe by Shoup in 2000 [Sho00]. We follow this approach and thus only define IB-KEMs.

**Definition 7.** An *identity-based key encapsulation mechanism* (IB-KEM) consists of the following four PPT algorithms:

- $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ takes as input the security parameter and outputs the public parameters $\mathsf{mpk}$ and the master secret key $\mathsf{msk}$.

- $\mathsf{usk}_{\mathsf{id}} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ returns the user secret key $\mathsf{usk}_{\mathsf{id}}$ for identity $\mathsf{id} \in \{0,1\}^\lambda$.
- $(\mathsf{ct}, \kappa) \xleftarrow{\$} \mathsf{Encap}(\mathsf{mpk}, \mathsf{id})$ returns a tuple $(\mathsf{ct}, \kappa)$, where $\mathsf{ct}$ is the ciphertext encapsulating $\kappa$ with respect to identity $\mathsf{id}$.
- $\kappa = \mathsf{Decap}(\mathsf{usk}_{\mathsf{id}}, \mathsf{ct}, \mathsf{id})$ returns the decapsulated key $\kappa$ or an error symbol $\perp$.

Correctness requires that for all $\lambda \in \mathbb{N}$, all $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{SetupIBK}(1^\lambda)$, all $\mathsf{id} \in \{0,1\}^\lambda$, all $(\kappa, \mathsf{ct}) \xleftarrow{\$} \mathsf{Encap}(\mathsf{mpk}, \mathsf{id})$ and all $\mathsf{usk}_{\mathsf{id}} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$:

$$\Pr[\mathsf{Decap}(\mathsf{usk}_{\mathsf{id}}, \mathsf{ct}, \mathsf{id}) = \kappa] \geq 1 - \mathsf{negl}(\lambda).$$

In order for the combination of an IB-KEM with a DEM to yield an adaptively-secure IBE scheme, we need that the IB-KEM has indistinguishable ciphertexts under adaptively chosen plaintext attacks, which we refer to as *IND-ID-CPA security*. We use the following standard IND-CPA-security notion for IB-KEMs from [BFMS08].

**Definition 8.** For an identity-based key encapsulation mechanism $\mathcal{IB}\text{-}\mathcal{KEM} = (\mathsf{SetupIBK}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with access (via oracle queries) to $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \cdot)$ let IND-ID-CPA be the security experiment depicted in Figure 2.2. We say that $\mathcal{A}$ is *legitimate*, if there is no $\rho_{\mathcal{A}} \in \{0,1\}^\lambda$ such that $\mathcal{A}_1$ or $\mathcal{A}_2$ query $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}^*)$, where $\mathsf{id}^*$ is the identity output by $\mathcal{A}_1$. We define the advantage of $\mathcal{A}$ in breaking the IND-ID-CPA security of $\mathcal{IB}\text{-}\mathcal{KEM}$ as

$$\mathsf{Adv}^{\mathsf{IND\text{-}ID\text{-}CPA}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) := \left|\Pr[G^{\mathsf{IND\text{-}ID\text{-}CPA}}_{\mathcal{IB}\text{-}\mathcal{KEM},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1] - 1/2\right|$$

We say that $\mathcal{IB}\text{-}\mathcal{KEM}$ is IND-ID-CPA $(t, \varepsilon)$-secure if $\mathsf{Adv}^{\mathsf{IND\text{-}ID\text{-}CPA}}_{\mathcal{A},\mathcal{IB}\text{-}\mathcal{KEM}}(\lambda) \leq \varepsilon$ for all adversaries probabilistic adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ running in time $t$.

## 2.4 Complexity Assumptions

Here, we introduce the complexity assumptions that we repeatedly use in Chapters 3 and 4. We will introduce further complexity assumptions in Chapter 5 in the immediate context where we need them.

The constructions of our VRFs in Chapter 3 and Chapter 4 are based on bilinear pairings. Informally, a bilinear pairing for a prime $p$ and two groups $\mathbb{G}_1, \mathbb{G}_2$ of size $p$ and a target group $\mathbb{G}_T$ also of size $p$ is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ such that for all generators $g_1$ of $\mathbb{G}_1$ and $g_2$ of $\mathbb{G}_2$ and all $a, b \in \mathbb{Z}_p$ is holds that

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}.$$

Bilinear pairings have more required properties that we formalize below. Moreover, note that we only consider so-called *Type-I* pairings. That is, pairings where $\mathbb{G}_1 = \mathbb{G}_2$. We refer to [GPS08] for a more detailed discussion of bilinear pairings.

We introduce (certified) bilinear group generators, which were originally described in [HJ16] to formalize how a pairing is chosen. The definition of (certified) bilinear group generators allow us to define complexity assumptions relative to the way the bilinear group is chosen end ensure that every group element has a unique encoding, which is required for the unique provability of the constructions of our VRFs.

**Definition 9.** A *Bilinear Group Generator* is a probabilistic polynomial-time algorithm $\mathsf{GrpGen}$ that takes as input a security parameter $\lambda$ (in unary) and outputs $\mathcal{BG} = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1)) \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$ such that the following requirements are satisfied.

1. $p$ is a prime and $\log(p) \in \Omega(k)$
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
   a) $(\mathbb{G}, \circ)$ and $(\mathbb{G}_T, \circ_T)$ form algebraic groups,
   b) $\phi$ is a group isomorphism from $(\mathbb{Z}_p, +)$ to $(\mathbb{G}, \circ)$ and
   c) $\phi_T$ 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,
$$x \neq 0 \Rightarrow e(\phi(x), \phi(x)) \neq \phi_T(0).$$

**Definition 10.** We say that group generator $\mathsf{GrpGen}$ is certified, if there exist deterministic polynomial-time (in the security parameter) algorithms $\mathsf{GrpVfy}$ and $\mathsf{GrpElemVfy}$ with the following properties.

**Parameter Validation.** Given the security parameter (in unary) and a string $\mathcal{BG}$, which is not necessarily generated by $\mathsf{GrpGen}$, algorithm $\mathsf{GrpVfy}(1^\lambda, \mathcal{BG})$ outputs 1 *if and only if* $\mathcal{BG}$ has the form
$$\mathcal{BG} = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1))$$
and all requirements from Definition 9 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 the security parameter (in unary) and two strings $\mathcal{BG}$ and $s$, $\mathsf{GrpElemVfy}(1^\lambda, \mathcal{BG}, s)$ outputs 1 if and only if $\mathsf{GrpVfy}(1^\lambda, \mathcal{BG}) = 1$, 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 $\mathcal{BG}$ to specify the representation of elements of $\mathbb{G}$.

Given the definition of (certified) bilinear group generators, we introduce the computational problems and complexity assumptions that we use in the constructions of VRFs. Note that for a finite group $\mathbb{G}$, we let

$$\mathbb{G}^* := \mathbb{G} \setminus \{1_{\mathbb{G}}\}$$

denote the group $\mathbb{G}$ without the identity element $1_{\mathbb{G}}$.

**Definition 11** ($q$-DDH problem)**.** For a bilinear group generator $\mathsf{GrpGen}$, let $\mathcal{BG} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$, $g, h \xleftarrow{\$} \mathbb{G}^*$, $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $T_0 := e(g,h)^{\alpha^{q+1}}$ and $T_1 \xleftarrow{\$} \mathbb{G}_T$. We then denote with

$$\mathsf{Adv}^{q\text{-}\mathsf{DDH}}_{\mathcal{B}}(\lambda) := \left|\Pr\left[\mathcal{B}(\mathcal{BG}, g, g^\alpha, \ldots, g^{\alpha^q}, h, T_0) = 1\right] - \Pr\left[\mathcal{B}(\mathcal{BG}, g, g^\alpha, \ldots, g^{\alpha^q}, h, T_1) = 1\right]\right|$$

the *advantage* of $\mathcal{B}$ in solving the $q$-DDH problem for groups generated by $\mathsf{GrpGen}$, where the probability is taken over the randomness of drawing $g, h, \alpha$ and $T$ and the internal randomness of $\mathcal{B}$. For functions $t : \mathbb{N} \to \mathbb{N}$ and $\varepsilon : \mathbb{N} \to [0,1]$, we say that $\mathcal{B}$ $(t,\varepsilon)$-solves the $q$-DDH problem relative to $\mathsf{GrpGen}$, if $\mathsf{Adv}^{q\text{-}\mathsf{DDH}}_{\mathcal{B}}(\lambda) \geq \varepsilon(\lambda)$ and $\mathcal{B}$ runs in time at most $t(\lambda)$.

**Definition 12** ($q$-DDH assumption)**.** Let $\mathsf{GrpGen}$ be a bilinear group generator. If for all polynomials $t : \mathbb{N} \to \mathbb{N}$ and all non-negligible functions $\varepsilon : \mathbb{N} \to [0,1]$, there is no algorithm $\mathcal{B}$ that $(t,\varepsilon)$-solves the $q$-DDH assumption relative to $\mathsf{GrpGen}$, then we say that the $q$-DDH problem is hard relative to $\mathsf{GrpGen}$.

Note that the $q$-DDH assumption is trivially implied by the decisional $q+1$ Bilinear Diffie-Hellman Exponent assumption introduced in [BBG05].

**Definition 13** ($q$-DBDHI problem, Definition 4 in [BB04a])**.** For a bilinear group generator $\mathsf{GrpGen}$, an algorithm $\mathcal{B}$ and $q \in \mathbb{N}$, let $\mathcal{BG} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$, $g, h \xleftarrow{\$} \mathbb{G}^*$, $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, $T_0 := e(g,h)^{1/\alpha}$ and $T_1 \xleftarrow{\$} \mathbb{G}_T$. We then denote with

$$\mathsf{Adv}^{q\text{-}\mathsf{DBDHI}}_{\mathcal{B}}(\lambda) := \left|\Pr\left[\mathcal{B}(\mathcal{BG}, g, h, g^\alpha, \ldots, g^{\alpha^q}, T_0) = 1\right] - \Pr\left[\mathcal{B}(\mathcal{BG}, g, h, g^\alpha, \ldots, g^{\alpha^q}, T_1) = 1\right]\right|$$

the *advantage* of $\mathcal{B}$ in solving the $q$-DBDHI problem for groups generated by $\mathsf{GrpGen}$, where the probability is taken over the randomness of $\mathsf{GrpGen}$, the random choices of $g, h$ and $\alpha$, and the internal randomness of $\mathcal{B}$. For functions $t : \mathbb{N} \to \mathbb{N}$ and $\varepsilon : \mathbb{N} \to [0,1]$, we say that $\mathcal{B}$ $(t,\varepsilon)$-solves the $q$-DBDHI problem relative to $\mathsf{GrpGen}$, if $\mathsf{Adv}^{q\text{-}\mathsf{DBDHI}}_{\mathcal{B}}(\lambda) \geq \varepsilon(\lambda)$ and $\mathcal{B}$ runs in time at most $t(\lambda)$.

**Definition 14** ($q$-DBDHI assumption)**.** Let $\mathsf{GrpGen}$ be a bilinear group generator. If for all polynomials $t : \mathbb{N} \to \mathbb{N}$ and all non-negligible functions $\varepsilon : \mathbb{N} \to [0,1]$, there is no algorithm $\mathcal{B}$ that $(t,\varepsilon)$-solves the $q$-DBDHI problem relative to $\mathsf{GrpGen}$, then we say that the $q$-DBDHI problem is hard relative to $\mathsf{GrpGen}$.

## 2.5 Sequence of Games Arguments

To prove the security of cryptographic schemes such as VRFs and IB-KEMs, modern cryptography uses reductions from complexity assumptions like the ones we introduced in Section 2.4 above to the security of the respective scheme. That is, the proof shows that any adversary that breaks the security of the cryptographic scheme implies the existence of an efficient algorithm for a computational problem. Since the complexity assumption states that such an algorithm can not exist, we thus conclude that the cryptographic scheme is secure under the respective complexity assumption.

These reductions can become rather complex and are thus prone to errors. To tame the complexity of such proofs and allow experts to efficiently verify the correctness, it is common to structure security proofs as so-called *sequence of games arguments*. Sequence of games arguments have been used known in cryptography for quite some time. An early example of their usage is Goldreich *et al.*'s work on the construction of pseudorandom functions [GGM86]. Since then, different notations and notions for sequence of games arguments have evolved. Most notable are the notations of *code-based games* by Bellare and Rogaway [BR06] and the *sequence of games* notation by Shoup [Sho04]. This thesis uses the latter notation by Shoup [Sho04].

A proof structured as a sequence of games argument makes small incremental changes to the security experiment. More specifically, consider one of the security experiments for VRFs or IB-KEMs, which we introduced in Section 2.3 and denote the event that the security experiment outputs 1 by $\mathsf{G}_0$. Thus, for the security of VRFs and IB-KEMs, we require that $\Pr[\mathsf{G}_0]$ is negligibly close to 1/2. We then define a sequence of games *Game 1* up to *Game* $n$, where there is an incremental change for each consecutive game. Denoting the event that Game $i$ outputs 1 by $\mathsf{G}_i$, these small incremental changes make it easy to relate $\Pr[\mathsf{G}_{i-1}]$ to $\Pr[\mathsf{G}_i]$ for all $i \in [n]$. In particular, the changes will be made such that, in the end, we will be able to conclude that

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_n]| \tag{2.1}$$

is negligible. Furthermore, Game $n$ will be constructed in a way such that it is easy to describe an algorithm $\mathcal{B}$ solving the computational problem and relating the $\mathcal{B}$'s success probability to the success probability of the adversary in Game $n$. This property, together with Equation (2.1) thus enable us to relate the adversary's advantage to the advantage of $\mathcal{B}$. In the proofs in this thesis, we will construct Game $n$ and Game $n-1$ such that $\Pr[\mathsf{G}_n] = 1/2$ and that $|\Pr[\mathsf{G}_n] - \Pr[\mathsf{G}_{n-1}]|$ is identical to the advantage of $\mathcal{B}$.

Over the years some useful tools for these types of proofs have been developed. One of the most common tool is the difference lemma. It is used to relate $\Pr[\mathsf{G}_{i-1}]$ and $\Pr[\mathsf{G}_i]$ if the challenger aborts the experiment in Game $i$ if an event $\mathsf{bad}$ happens. Informally, the lemma then states that $|\Pr[\mathsf{G}_{i-1}] - \Pr[\mathsf{G}_i]|$ is at most $\Pr[\mathsf{bad}]$. We provide the formal statement below and refer to [Sho04, Lemma 1] for the proof.

**Lemma 1** (Difference lemma [Sho04, Lemma 1])**.** *Let $A, B$ and $F$ be events in some probability distribution, and suppose that $A \wedge \neg F \iff B \wedge \neg F$. Then*

$$|\Pr[A] - \Pr[B]| \leq \Pr[F]$$

*holds.*

## 2.6 Partitioning Arguments for Adaptive Security

As the final preliminary before presenting the results of the thesis, we introduce *partitioning arguments*, a proof strategy that we will make use of repeatedly throughout this thesis. Thus, we first describe partitioning arguments and then discuss the challenges that come with using them to prove the pseudorandomness of VRFs or the IND-ID-CPA security of IB-KEMs.

Consider IB-KEMs as a concrete example, but note that all statements also apply to VRFs. A reduction that uses a partitioning argument, at the very beginning, partitions the identity space in two disjoint sets as follows:

1. A *"controlled set"*, which contains all identities for which the reduction is able to answer queries made by the adversary. However, if the adversary chooses the challenge identity in the controlled set, the reduction is not able to extract a solution to the underlying complexity assumption.

2. A *"uncontrolled set"*, which contains all identities for which the reduction is able to extract a solution to the underlying complexity assumption if the adversary chooses it as the challenge. However, the reduction is not able to answer any queries for identities that lies in this subset.

Thus, if the adversary makes a query for any identity in the uncontrolled set or chooses the challenge identity in the controlled set, then the reduction can only abort and output a uniformly random bit. Thus, the reduction has to choose the controlled set and the uncontrolled set in a probabilistic way such that it does not has to abort with a non-negligible probability. More specifically, all queries $\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}$ have to fall into controlled set and the challenge identity $\mathsf{id}^*$ has to fall in the uncontrolled set a non-negligible probability.

Partitioning is often used in the random oracle model, which we discussed in Section 1.2.3. For instance, the well-known security proofs of Full-Domain Hash signatures [BR96], BLS signatures [BLS04], or the Boneh-Franklin IBE [BF03] use this approach. Furthermore, partitioning is the only known way to prove security of unique signatures[1] or VRFs against *adaptive adversaries*.

However, this thesis is concerned with partitioning arguments that do not use the random oracle heuristic, *i.e.*, proofs in the standard model. There are several semi-generic techniques for partitioning arguments in the standard model. Most notably

[1] That is, digital signatures where for any given (public key, message)-pair there exists only one unique string that is accepted as a signature by the verification algorithm.

are *admissible hash function*s (AHFs) [BB04a] and *programmable hash function*s (PHFs) [HK08, HK12]. Admissible hash functions are a combinatoric approach to choose the controlled and uncontrolled set and we discuss this approach in detail in Section 3.2. In contrast to AHFs, programmable hash functions also describe how the partitioning in controlled and uncontrolled sets are embedded into a construction. While Hofheinz and Kiltz originally defined PHFs for schemes based on the RSA assumption, the discrete logarithm assumption and closely related assumptions [HK08, HK12], Zhang *et al.* also introduce PHFs for constructions based on lattice problems [ZCZ16]. We introduce more efficient constructions of PHFs for lattices in Chapter 5.

Partitioning arguments for decisional security. We use partitioning arguments to prove the pseudorandomness of VRFs (see Definition 6) and the IND-ID-CPA security of IB-KEMs (see Definition 8). Both these primitives with their accompanying security notions have in common that they require *decisional security*, meaing the adversary has to correctly guess a bit $b$ with a probability that is non-negligibly far away from $1/2$ to break the security. Consequently, in all our constructions, a reduction from a decisional assumption like the $q$-DBDHI assumption (see Definition 14) or the $q$-DDH assumption (see Definition 12) are used to prove the security.

This comes with the issue that the event that the reduction aborts can be correlated with the choice of the queries and the challenge chosen by the adversary. More specifically, let succ-red be the event that the reduction solves the complexity assumption and let abort be the event that the reduction aborts and outputs a random bit. For clarity in this informal description, we assume that the reduction always outputs the correct bit when the adversary succeeds and the reduction does not abort. We then have that

$$\begin{aligned}\Pr[\mathsf{succ\text{-}red}] &= \Pr[\mathsf{succ\text{-}red} \wedge \mathsf{abort}] + \Pr[\mathsf{succ\text{-}red} \wedge \neg\mathsf{abort}] \\ &= \frac{1}{2}(1 - \Pr[\neg\mathsf{abort}]) + \Pr[\mathsf{succ\text{-}red} \wedge \neg\mathsf{abort}] \\ &= \frac{1}{2} + \Pr[\mathsf{succ\text{-}red} \wedge \neg\mathsf{abort}] - \frac{\Pr[\neg\mathsf{abort}]}{2}.\end{aligned}$$

Thus, the reduction has to ensure that not only $\Pr[\neg\mathsf{abort}]$ is large enough, but also that $\Pr[\neg\mathsf{abort}]$ is not too large. Waters achieved this by introducing an *artificial abort* step [Wat05]. In this step, the reduction uniformly at random samples polynomially many query potential sequences and by that estimates the probability $\tau$ to abort for a uniformly random query sequence. The reduction then aborts and outputs a random bit with a probability based on $\tau$ and by that makes the event abort, not technically, but virtually independent from the particular sequence of queries the adversary chose and thus allows the reduction to succeed.

The artificial abort technique has the major downside that the estimation of $\tau$ is computationally quite expensive. Bellare and Ristenpart addressed this issue

by proving close upper and lower bounds on $\Pr[\neg\mathsf{abort}]$ in a technically involved proof [BR09a]. Ultimately, this allowed them to avoid the computationally expensive artificial abort step that Waters required.

Throughout the thesis, we discuss further approaches to prove adaptive security using partitioning arguments. In Chapters 3 and 5, we discuss partitioning arguments that yield particularly efficient constructions and in Chapter 4, we present a partitioning argument that makes the event $\mathsf{abort}$ independent from the sequence of queries chosen by the adversary.

# 3 Efficient Verifiable Random Functions

In this chapter, we study the construction of efficient verifiable random function from bilinear pairings in the standard model. To this end, we first study *admissible hash function*s (AHFs) as a common building block to construct VRFs in the standard model and derive techniques to show the inherent limitations of AHFs in Section 3.2. Motivated by these limitations, we proceed in Section 3.3, by presenting *computational admissible hash function*s (cAHFs) and their instantiation from *truncation-collision resistant* hash functions, first introduced in [JK18], as a drop-in replacement for AHFs and show how cAHFs can be used to construct efficient VRFs. In Section 3.4, we introduce *blockwise partitioning* from *weak near-collision resistant* hash functions in order to construct even more efficient VRFs than the VRFs we constructed from cAHFs. However, this gain in efficiency comes at the cost that blockwise partitioning can not replace AHFs as generically as cAHFs can. In addition to the VRF built from blockwise partitioning we present in this chapter, blockwise partitioning can also be used to construct efficient identity-based key encapsulation mechanisms. We present an identity-based key encapsulation mechanism using blockwise partitioning in Chapter 5. Finally, we compare VRFs from pairings in detail in Section 3.5 and conclude the chapter in Section 3.6.

Author's contributions. Overall, the results in this chapter are based on joint work with Tibor Jager and Rafael Kurek published in [JKN21, JN19a]. More specifically, the results presented in Section 3.2 are based on [JN19a] and are the author's contribution. The results presented in Section 3.3 are also based on [JN19a] but are due to joined work with Tibor Jager. In particular the VRF presented in Construction 1 is a more efficient variant of the VRF published in [JN19a]. It is also described in the full version [JN19b] of [JN19a]. Furthermore, the results presented in Section 3.4 are based on the joint work [JKN21]. However, the notions of weak near-collision resistance and blockwise partitioning are mostly the authors contribution. Moreover, the VRF presented in Construction 2 and Lemma 7 and the accompanying proofs are mostly due to myself. The detailed comparison of VRFs in Section 3.5, except for the discussion of Kohl's VRF were published in [JN19a] and are mostly due to myself.

## 3.1 Motivation and Overview

VRFs have found a wide range of applications in theory and in practice. One of the most notable ones is the recent application of VRFs in *proof of stake* consensus mechanisms, like the ones used in the Algorand Blockchain [GHM+17], the Cardano Blockchain [BGK+18, DGKR18] and the DFINITY Blockchain [AMNR18]. Further applications are in *key transparency systems* like CONIKS [MBB+15], the novel Merkle$^2$ [HHK+21] system, or the discontinued Google key transparency [MGZ+18], where VRFs prevent the enumeration of all users that have keys in the system. Similarly, VRFs are used in the by now inactive *DNSSEC extension NSECv5* [VGP+18] proposal, where they provably prevent zone enumeration attacks in the authenticated denial of existence mechanism of DNSSEC [GNP+15]. Further classical applications are resettable zero-knowledge proofs [MR01], lottery systems [MR02], verifiable transaction-escrow systems [JS04], updatable zero-knowledge databases [Lis05] and E-Cash [ASM07, BCKL09]. The wide range of applications has led to currently ongoing efforts to standardize VRFs [GRPV21]. These VRFs are efficient, but the accompanying security proofs rely on the random oracle heuristic [BR93], which can not be instantiated in general [CGH04].

Admissible hash functions. AHFs are a generic and useful tool to enable partitioning proofs in the *standard model*, that is, without random oracles. They thus enable construction with adaptive security in the standard model. AHFs were formally introduced in [BB04b] but had implicitly already been used by Lysyanskaya [Lys02]. They are a ubiquitous tool in public-key cryptography and have been used to realize numerous cryptographic primitives with strong adaptive security and without random oracles, such as unique signatures [Lys02], verifiable random functions [Bit17a, HJ16, Jag15, Lys02], different variants of identity-based encryption [AFL12, BB04b], Bonsai trees [CHKP12], programmable hash functions [FHPS13], and constrained PRFs [AMN+19, DKN+20]. All recent constructions of VRFs that focus on efficiency, such as [Jag15, HJ16, Yam17a, Kat17, Koh19], rely on AHFs.

Practical Instantiability of AHFs. Given the large number of cryptographic constructions based on AHFs and their relevance for efficient VRFs in the standard model, it is interesting and important to ask how AHF-based constructions can be instantiated in *practice*. Known constructions are based on different types of error-correcting codes with suitable minimal distance, which is required to be a *constant* fraction of the length of the code, in order to make the partitioning argument go through with noticeable success probability. There are many possible codes to choose from [Gil52, Jus72, SS96, Var57, Zém01], which yield very different concrete instantiations with very different efficiency and security properties.

Most aforementioned works mention that one or another of these error-correcting codes can be used to instantiate their AHFs in the *asymptotic* setting, but it is never

clarified how their constructions can be instantiated *concretely*, by explaining how the underlying code and other cryptographic parameters must be chosen, taking into account the considered security parameter, deployment parameters such as the number of AHF evaluations by a realistic adversary, and the tightness of the security proof. The concrete choice of the code used to instantiate the AHF has a very significant impact on the efficiency of the resulting cryptosystem. Hence, while AHFs provide a powerful generic tool to achieve provable security *asymptotically*, it is completely unclear how efficiently they can be instantiated *concretely*.

Contributions. We make progress regarding most of the issues laid out above by first developing tools to assess the practicality of AHFs and then providing more efficient alternatives from non-standard yet plausible assumptions. We discuss the contributions in more detail below.

- We assess how AHFs can be instantiated with *error-correcting code*s. We show that while AHFs are theoretically sufficient to obtain polynomial-time constructions and security against polynomial-time adversaries in the asymptotic setting, they yield only extremely inefficient concrete instantiations. By applying bounds on error-correcting codes from classical coding theory, we point out inherent limitations of concrete instantiations of the AHFs presented in prior work. Concretely, we show that even with codes that meet the *Gilbert-Varshamov* (GV) or *McEliece-Rodemich-Rumsey-Welch* (MRRW) bound, even optimized variants [Kat17, Yam17a] of known verifiable random functions [Jag15] have only very inefficient practical instantiations.

- Standard AHFs based on error-correcting codes are essentially an *information-theoretic* primitive, which works unconditionally and even for computationally unbounded adversaries, which of course is stronger than necessary for most applications. We introduce computational admissible hash functions, which relax this requirement in the sense that they are only required to partition the considered set successfully in the presence of a *computationally bounded* adversary. This approach makes it possible to overcome the aforementioned limitations of AHFs. We also give a concrete instantiation of cAHFs, based on the notion of *truncation-collision resistant hash functions* from [JK18]. We furthermore describe a new efficient VRF, based on Jager's VRF [Jag15] that uses cAHFs. It thus showcases how cAHFs can be applied in constructions and proofs and can semi-generically replace AHFs.

- We introduce *blockwise partitioning* as a new approach to leverage the assumption that a cryptographic hash function is *weak near-collision resistant*. In Section 3.4, we show that our technique yields a VRF that achieves both small public keys and small proofs simultaneously. Furthermore, the size of the keys and proofs is not only asymptotically small but also concretely. As we see in our detailed construction in Section 3.5, other constructions, like [Yam17a, Kat17], that also achieve (poly-)logarithmic size public keys or

| Schemes | $\lvert\mathsf{vk}\rvert$ | $\lvert\pi\rvert$ | Assumption | Security loss |
|---|---|---|---|---|
| [HW10] | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda \cdot Q)$-DDHE | $\mathcal{O}(\lambda Q/\varepsilon)$ |
| [BMR10b] | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$-DDH | $\mathcal{O}(\lambda Q/\varepsilon)$* |
| [Jag15] | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\log(Q/\varepsilon))$-DDH | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| [HJ16] | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$ | DLIN | $\mathcal{O}(\lambda \log(\lambda) Q^{2/c}/\varepsilon^3)$ |
| [Yam17a] Sec. 6.1 | $\omega(\lambda \log^2 \lambda)^\dagger$ | $\omega(\log^2 \lambda)^\dagger$ | $\tilde{\mathcal{O}}(\lambda)$-DBDHI | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| [Yam17a] Sec. 6.2 | $\omega(\log^2 \lambda)^\dagger$ | $\omega(\sqrt{\lambda} \log^2 \lambda)^\dagger$ | $\tilde{\mathcal{O}}(\lambda)$-DBDHI | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| [Yam17b] App. C. | $\omega(\log^2 \lambda)^\dagger$ | $\mathsf{poly}(\lambda)$ | $\mathsf{poly}(\lambda)$-DBDHI | $\mathcal{O}(\lambda^2 Q/\varepsilon^2)$ |
| [Kat17] Sec. 5.1 | $\omega(\log^2 \lambda)^\dagger$ | $\omega(\lambda \log^2 \lambda)^\dagger$ | $\omega(\log^2 \lambda)^\dagger$-DBDHI | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| [Kat17] Sec. 5.3 | $\omega(\sqrt{\lambda} \log \lambda)^\dagger$ | $\omega(\log \lambda)^\dagger$ | $\omega(\log^2 \lambda)^\dagger$-DBDHI | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| [Ros18] | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$ | DLIN | $\mathcal{O}(\lambda \log(\lambda) Q^{2/c}/\varepsilon^3)$ |
| [Koh19] | $\omega(\lambda \log \lambda)^\dagger$ | $\omega(\log \lambda)^\dagger$ | DLIN | $\mathcal{O}(\lvert\pi\rvert \log(\lambda) Q^{2/\nu}/\varepsilon^3)$ |
| [Koh19] | $\omega(\lambda^{2+2\eta})$ | $\omega(1)^\dagger$ | DLIN | $\mathcal{O}(\lvert\pi\rvert \log(\lambda) Q^{2+2/\nu}/\varepsilon^3)$ |
| $\mathcal{VRF}_{\mathsf{cAHF}}$ in Construction 1 | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\lambda)$ | $\mathcal{O}(\log \lambda)$-DDH | $\mathcal{O}(t^2/\varepsilon)$ |
| $\mathcal{VRF}_{\mathsf{Blk}}$ in Construction 2 | $\mathcal{O}(\log \lambda)$ | $\mathcal{O}(\log \lambda)$ | $\mathcal{O}(t^2/\varepsilon)$-DBDHI | $\mathcal{O}(t^2/\varepsilon)$ |

*Table 3.1:* We compare adaptively-secure VRF schemes in the standard model. We measure the size of $\mathsf{vk}$ and $\pi$ in the number of the respective group elements and the size of $\mathsf{sk}$ as the number of $\mathbb{Z}_p$ elements. $Q, \varepsilon$ and $t$ respectively denote the number of queries an adversary makes, the adversaries advantage against the security of the respective VRF and the adversaries runtime. Most of the constructions use an error correcting code $C : \{0,1\}^\lambda \to \{0,1\}^n$ with constant relative minimal distance $c \leq 1/2$, where $n$,$\nu > 1$ can be chosen arbitrarily close to 1 by choosing $c$ arbitrarily close to $1/2$ [Gol08, Appendix E.1]. However, this leads to larger $n$ and by that to larger public keys and/or proofs.

† Note that these terms only hold for "$\lambda$ large enough" and therefore, key and proof sizes might have to be adapted with larger constants in order to guarantee adequate security.

* The loss we state for [BMR10b] is as in the recently updated full version [BMR10a].

proofs suffer from large constant factors, even under very optimistic assumptions. Furthermore, we will show in Chapter 5 how blockwise partitioning enables the construction of efficient IB-KEMs from lattices. Moreover, my co-authors Tibor Jager and Rafael Kurek show in our joined [JKN21] work that blockwise partitioning also yields highly efficient IB-KEMs from bilinear pairings.

We asymptotically compare our construction with previous constructions in Table 3.1, which is based on the respective table by Kohl [Koh19].

Related work. Boneh and Boyen [BB04b] formally introduced AHFs to construct identity-based encryption schemes without random oracles. *Balanced* AHFs were introduced in [Jag15]. The balancedness makes it possible to apply AHF-based partitioning directly in security proofs considering "indistinguishability-based" se-

curity experiments, without requiring the *artificial abort* approach, which was introduced by Waters [Wat05]. Balanced AHFs were used to construct verifiable random functions [Bit17a, HJ16, Jag15, Yam17a], IBE [Yam17a], constrained pseudorandom functions [AMN+19, DKN+20], and distributed PRFs [LST18]. We consider both standard and balanced AHFs in detail in Section 3.2.

Recently, Yamada, Katsumata and Kohl developed techniques to optimize VRFs based on AHFs. Yamada [Yam17a] and Katsumata [Kat17] presented VRFs that encode the information of the "controlled" set into shorter bit strings and employ the AHF on this shorter string. Kohl [Koh19] applied a similar approach to the VRF construction of [HJ16] to obtain a VRF with very shorter proofs that is secure under standard assumptions in the standard model.

Most previous applications of AHFs consider a setting where a polynomially-bounded number of $Q$ elements $X^{(1)}, \ldots, X^{(Q)}$ must fall into the "controlled" set, while *one* "challenge" element $X^*$ must fall into the "uncontrolled" set for the reduction in the security proof to be successful. This matches what is required for most common security experiments for primitives such as digital signatures, VRFs, IBE, and many others. Chen *et al.* [CHZ16] generalize this to AHFs that can handle more than one challenge element and ensure that $n > 1$ challenge elements $X^{(1)*}, \ldots, X^{(n)*}$ fall into the "uncontrolled" set, and give a construction with $n = 2$.

AHFs are related to *programmable hash functions* (PHFs) [HJK11, HK12], but are more general, in the sense that PHFs can generically be constructed from AHFs, but there exist cryptographic primitives, such as VRFs, for which only constructions based on AHFs are known to exist, but not on PHFs.

Moreover, Zhandry introduced *extremely lossy function*s (ELFs) [Zha16, Zha19a, Zha19b]. These are hash functions that allow the reduction to choose the hash functions image size depending on the adversary, such that a function with a small image size is indistinguishable from an injective hash function. This enables a similar reduction strategy as AHFs do. We discuss ELFs in more detail in the context of blockwise partitioning and weak near-collision resistance in Section 3.4.

## 3.2 Admissible Hash Functions and their Limitations

*Admissible hash function*s (AHFs) are a ubiquitous tool to achieve adaptive security in cryptographic constructions by enabling a partitioning argument as discussed in Section 2.6. In this chapter, we first formally introduce AHFs, show how they can be instantiated from error-correcting codes and finally show to what extent known instantiations are limited in their real-world applicability due to bounds from coding theory.

High level perspective. In order to sketch the main idea, consider VRFs and their security experiment $G^{\mathsf{Psrd}}$ (see Definition 6) as an example. Denote by $X^{(1)}, \ldots, X^{(Q)} \in \{0,1\}^\lambda$ all the queries made by the adversary and let $X^* \in \{0,1\}^\lambda$ be the challenge chosen by the adversary. An AHF is a function $\mathsf{AHF}$ that maps

each input to a longer bit string and amplifies differences in the inputs. This amplification then allows the reduction to guess a carefully chosen number $\eta$ of the bits of $\mathsf{AHF}(X^*)$. Due to the amplification of differences and the requirement in the game $G^{\mathsf{Psrd}}$ from Definition 6 that $X^{(i)} \neq X^*$ has to hold for all $i \in [Q]$, the reduction's guess is correct only for the $\mathsf{AHF}(X^*)$ but not for any $X^{(i)}$. To the best of our knowledge, this amplification is always achieved by using an error-correcting code.

### 3.2.1 Defining Admissible Hash Functions

AHFs have, to the best of our knowledge, first been used by Lysyanskaya in [Lys02] but have been formally introduced by Boneh and Boyen in [BB04b]. Since then, many different definitions of AHFs have emerged. Below, we choose a definition we deem best suited to capture the core idea and most applications of AHFs such as for example those in [AFL12, FHPS13, Kat17].

**Definition 15** (Admissible Hash Function [BB04b])**.** Let $n, k : \mathbb{N} \to \mathbb{N}$ be polynomially bounded functions and let $\mathcal{AHF} := \{\mathsf{AHF}^\ell : \{0,1\}^{k(\ell)} \to \{0,1\}^{n(\ell))}\}_{\ell \in \mathbb{N}}$ be a family of functions. For all $\ell \in \mathbb{N}$, $\mathsf{K} \in \{0,1,\bot\}^{n(\lambda)}$ and $X \in \{0,1\}^{k(\lambda)}$ we then let

$$\mathsf{F}_{\mathsf{K},\ell}(X) := \begin{cases} 1, & \text{if } \forall\, i \in [n(\ell)] : \mathsf{AHF}^\ell(X)_i = \mathsf{K}_i \vee \mathsf{K}_i = \bot \text{ holds and} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

We say that $\mathcal{AHF}$ is a family of *admissible hash function*s (AHFs) if there exists a PPT $\mathsf{AdmSmp}(1^\lambda, \cdot)$ outputting $\mathsf{K} \in (\{0,1\} \cup \bot)^{n(\lambda)}$ such that for all polynomials $Q : \mathbb{N} \to \mathbb{N}$, there is a non-negligible function $\gamma_{\min} : \mathbb{N} \to [0,1]$ such that for all $(X^{(1)}, \ldots, X^{(Q(\lambda))}, X^*) \in (\{0,1\}^{k(\lambda)})^{Q(\lambda)+1}$ with $X^* \neq X^{(i)}$ for all $i \in [Q(\lambda)]$, it holds that

$$\gamma_{\min}(\lambda) \leq \Pr\left[\mathsf{F}_{\mathsf{K},\lambda}\left(X^{(1)}\right) = \cdots = \mathsf{F}_{\mathsf{K},\lambda}\left(X^{(Q(\lambda))}\right) = 0 \wedge \mathsf{F}_{\mathsf{K},\lambda}\left(X^*\right) = 1\right], \tag{3.2}$$

where the probability is over the choice of $\mathsf{K} \xleftarrow{\$} \mathsf{AdmSmp}(1^\lambda, Q(\lambda))$.

Informally, a family of AHFs thus guarantees that if all messages, identities, VRF inputs or respective entities for the considered primitive are passed trough $\mathsf{AHF}^\lambda$, then the reduction can always partition the respective space in a controlled and uncontrolled set such that the partitioning succeeds with a non-negligible probability. That is, with a non-negligible probability, the reduction can answer all queries made by the adversary and can extract a solution to the underlying complexity assumption if the adversary wins the security experiment.

Variants of AHFs. Since there have been many different definitions of AHFs, we provide a small overview over the decisions we made in our definition of AHFs. Most notably, we limited ourselves to families of admissible hash functions that

take binary inputs and produce binary outputs. In contrast, some definitions consider functions that map binary inputs to tuples over some abstract finite alphabet [BB04b, AFL12, FHPS13]. As Bitansky has shown, AHFs with non-binary outputs can also be constructed [Bit20]. Moreover, Kohl shows how this can be used to construct efficient VRFs [Koh19]. However, since most applications of AHFs require that the outputs of an AHF are binary, this is a only a minor limitation. Furthermore, we discuss Kohl's VRF in detail in Section 3.5.

Another decision we made in the definition above is to have only a single function $\mathsf{AHF}^\lambda$ for each potential value of the security parameter instead of a *family of functions*, as for example in [BB04b, AFL12]. These latter works define AHFs as a family of families of functions because they include the application of a hash function $H$ from a family of collision resistant hash functions. $H$ is then evaluated before what we refer to as an AHF is evaluated on the output of $H$. This is also why Boneh and Boyen chose to name the primitive admissible *hash* function [BB04b]. Nonetheless, most applications of AHFs and in particular more recent ones, *e.g.* those in [Bit17a, Bit20, FHPS13, HJ16, Jag15, Kat17, Koh19, Yam17a], assume that a collision resistant hash function is used in advanced to process arbitrary length inputs. For this reason, we define AHFs with a bounded length input space and view the application of a collision resistant hash function as in [BB04b] only as a generic way of processing arbitrary length inputs with a purely information-theoretic AHF.

AHFs and decisional security experiments. Intuitively, admissible hash functions guarantee a lower bound on the probability that the partitioning argument succeeds. This is sufficient to prove the security of primitives where the adversary has to solve a computational problem such as digital signatures. However, when the adversary has to solve a decisional problem to break the security of a primitive as for example for VRFs, IB-KEMs for constrained pseudorandom functions, this is not sufficient. As we discussed in Section 2.6 this is because the event that the adversary is successful is not independent from the event hat the partitioning argument is successful. Using the *artificial abort technique* from Waters [Wat05], AHFs can be made to work also in this decisional context, see for example [AFL12, HJ16, Koh19]. Unfortunately, the artificial abort technique comes at the cost of a substantial computational overhead in the reduction. Due to this undesirable property, Bellare and Ristenpart further investigated the issue. They showed that the computational overhead of the artificial abort technique can be avoided if, in addition to the lower bound $\gamma_{\min}(\lambda)$ on the probability for the partitioning argument to succeed, there is also an upper bound $\gamma_{\max}(\lambda)$ that is close enough to the lower bound. The concrete requirements to apply Bellare's and Ristenpart's technique are well captured by the following Lemma by Katsumata and Yamada [KY16, Lemma 8].

**Lemma 2** ([KY16, Lemma 8], implicit in [BR09a])**.** *For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the $G^{\mathsf{IND\text{-}ID\text{-}CPA}}$ (see Definition 8) or the $G^{\mathsf{Psrd}}$ (see Definition 6) security experiment that has advantage $\varepsilon : \mathbb{N} \to [0, 1]$, let us denote with $b \in \{0, 1\}$ the bit chosen*

*by the security experiment and with* $b' \in \{0,1\}$ *the bit output by* $\mathcal{A}$ *in the end. Further, let* $\mathcal{Q}^* = \{X^{(1)}, \ldots, X^{(Q)}, X^*\}$ *be the identities or VRF inputs for which* $\mathcal{A}$ *queried its oracle and the challenge identity or challenge VRF input, respectively. For a function* $\gamma$ *that maps* $\mathcal{Q}^*$ *to a value in* $[0,1]$*, let* $\widehat{b} := b'$ *with probability* $\gamma(\mathcal{Q}^*)$ *and let* $\widehat{b} \xleftarrow{\$} \{0,1\}$ *with probability* $1 - \gamma(\mathcal{Q}^*)$*. It then holds that*

$$\left| \Pr\left[b = \widehat{b}\right] - \frac{1}{2} \right| \geq \varepsilon(\lambda) \cdot \gamma_{\min}(\lambda) - \frac{\gamma_{\max}(\lambda) - \gamma_{\min}(\lambda)}{2},$$

*where* $\gamma_{\min} : \mathbb{N} \to [0,1]$ *and* $\gamma_{\max} : \mathbb{N} \to [0,1]$ *are upper and lower bounds on a* $\gamma(\mathcal{Q}^*)$ *taken over all possible* $\mathcal{Q}^*$*.*

Informally, the lemma guarantees that if $\gamma_{\min}$ is non-negligible and $\gamma_{\max}$ and $\gamma_{\max}$ are sufficiently close, then a reduction that aborts and outputs a random bit whenever the partitioning fails still guesses the correct bit $b$ correctly with a non-negligible probability. We do not provide a proof for Lemma 2 and refer to [BR09a] for a detailed exposition of the proof.

*Remark* 3. We explicitly state the applicability of Bellare's and Ristenpart's technique to VRFs and IB-KEMs because these are the primitives this thesis is focussed on. Nonetheless, the technique is also applicable to further primitives such as identity-based encryption [BR09a] or constrained pseudorandom functions [DKN+20].

Balanced admissible hash function. To use Bellare's and Ristenpart's technique together with AHFs, Jager introduced *balanced admissible hash function*s (bAHFs) [Jag15, Definition 5], which guarantee close lower and upper bounds as required to apply Lemma 2.

**Definition 16** (Balanced admissible hash functions [Jag15, Definition 5])**.** Let $\mathcal{AHF}$ and $\mathsf{F}$ be as in Definition 15. We call $\mathcal{AHF}$ a family of *balanced admissible hash function* (bAHF) if there exists a PPT $\mathsf{K} \xleftarrow{\$} \mathsf{AdmSmp}(1^\lambda, \cdot, \cdot)$ with $\mathsf{K} \in (\{0,1\} \cup \{\bot\})^{n(\lambda)}$ such that for all polynomials $Q : \mathbb{N} \to \mathbb{N}$ and all non-negligible functions $\varepsilon : \mathbb{N} \to [0,1]$ there are functions $\gamma_{\max}, \gamma_{\min} : \mathbb{N} \to [0,1]$ such that for all $(X^{(1)}, \ldots, X^{(Q(\lambda))}, X^*) \in (\{0,1\}^{k(\lambda)})^{Q(\lambda)+1}$ with $X^* \neq X^{(i)}$ for all $i \in [Q(\lambda)]$ it holds that

$$\gamma_{\min}(\lambda) \leq \Pr\left[\mathsf{F}_{\mathsf{K},\lambda}\left(X^{(1)}\right) = \cdots = \mathsf{F}_{\mathsf{K},\lambda}\left(X^{(Q)}\right) = 0 \wedge \mathsf{F}_{\mathsf{K},\lambda}\left(X^*\right) = 1\right] \leq \gamma_{\max}(\lambda), \tag{3.3}$$

and that

$$\tau(\lambda) := \varepsilon(\lambda)\gamma_{\min}(\lambda) - \frac{\gamma_{\min}(\lambda) - \gamma_{\max}(\lambda)}{2} \tag{3.4}$$

is a non-negligible function, where the probability is over the choice of the key of the AHF $\mathsf{K} \xleftarrow{\$} \mathsf{AdmSmp}(1^\lambda, Q(\lambda), \varepsilon(\lambda))$.

### 3.2.2 Instantiating Admissible Hash Functions

After we discussed different notions and properties of AHFs in the previous section, we now consider their instantiation. Since, to the best of our knowledge, all instantitations of AHFs are based on *error-correcting code*s (ECCs), we first formally introduce ECCs and then describe how AHFs can be instantiated using ECCs.

#### Error Correcting Codes.

The original purpose of *error-correcting code*s (ECCs) is to encode messages such that if an encoded messages, which referred to as *code word*, is transmitted over an unreliable medium and errors are introduced, the original message can uniquely be identified. ECCs achieve this by ensuring that all code words potentially produced by an ECC differ in many positions. This difference is also referred to as the *Hamming distance* and we formally introduce it below.

**Definition 17** (Hamming weight and distance)**.** Let $n \in \mathbb{N}$, $q$ a prime power and $x, y \in \mathbb{F}_q^n$, then $\mathrm{wt}(x)$ is defined as the number of components of $x$ that are not zero. We call $\mathrm{wt}(x)$ the *Hamming weight of* $x$ and $\Delta(x, y) := \mathrm{wt}(x - y)$ the *Hamming distance* between $x$ and $y$.

Given the definition of the hamming distance, we define ECCs as follows.

**Definition 18** (Error-correcting code)**.** Let $q$ be a prime power and let $k, n \in \mathbb{N}$ with $k < n$. We then refer to a function $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ as a $q$-ary *error-correcting code* (ECC). We say that $C$ has *minimal distance*

$$d := \min_{\substack{x,x' \in \mathbb{F}_q^k \\ x \neq x'}} (\Delta(C(x), C(x'))).$$

Furthermore we refer to $\delta(C) := d/n$ as the *relative minimal distance* and to $\mathcal{R}(C) := k/n$ as the *rate* of $C$. If $C$ has a minimal distance of $d$, we say that $C$ is an $[n, k, d]_q$ ECC. If there is a matrix $G \in \mathbb{F}_q^{k \times n}$ such that $C(x) = xG$ for all $x \in \mathbb{F}_q^k$, then we say that $C$ is a *linear q-ary error-correcting code*.

Informally, the *rate* of an ECC captures how much longer the code words are compared to the messages. Thus, a small rate is desirable in practice to not blow up the size of the transmitted data more than necessary. Furthermore, the *minimal distance* of an error correcting code is in indicator on how many errors an ECC can at least correct. That is, for an ECC with a minimal distance of $d \in \mathbb{N}$ it is always possible to recover the original message if not more than $\lfloor 1/2(d-1) \rfloor$ errors were introduced during transmission [MS98, Theorem 2]. Even though we don't use ECCs in this thesis to ensure the correct transmission of data, we will see that both these qualities of ECCs significantly affect the efficiency of cryptographic schemes using AHFs based on ECCs.

*Remark* 4. The ECCs we introduced above are more specifically known as *block codes*. Other types of codes such as convolutional codes [HP03, Chapter 14] are also known. Nonetheless, we limit our attention to block codes because all instantiations of AHFs known to us are based on block codes. Furthermore, coding theorists sometimes also use a slightly more general notion of ECCs, which defines a code to be any subset $C \subseteq \mathbb{F}_q^{k \times n}$. However, Definition 18 is better suited to our purposes because only ECCs with an efficient encoding algorithm are relevant to this thesis.

### Instantiating Admissible Hash Functions with Error Correcting Codes.

To the best of our knowledge, all instantiations of AHFs and bAHFs are based on families of ECCs with a constant relative minimal distance $\delta > 0$, meaning all codes in the family have a relative minimal distance of at least $\delta$. An example for such a family of codes are the *Justesen codes* [Jus72]. In addition to a constant relative minimal distance, ECCs in this family of codes have a constant rate, meaning that the length $n$ of the code words is linear in the length $k$ of the encoded message.

Furthermore, all instantiations of AHFs known to us use an algorithm $\mathsf{AdmSmp}$ that samples $\mathsf{K}$ as follows. First, for all $m \in [n(\lambda)]$, let $\{0,1\}_{\perp}^{(n(\lambda),m)}$ be the subset of $\{0,1,\perp\}^{n(\lambda)}$ whose elements have *exactly* $m$ positions that are *not* $\perp$. Then $\mathsf{AdmSmp}$ calculates $\eta \in [n(\lambda)]$ as a function of $Q(\lambda)$ and the advantage $\varepsilon(\lambda)$ of a given adversary $\mathcal{A}$ (the latter only for bAHFs). Finally, $\mathsf{AdmSmp}$ chooses $\mathsf{K}$ uniformly at random from $\{0,1\}_{\perp}^{n(\lambda),\eta)}$. We formalize these statements below. That is, we first prove generic bounds $\gamma_{\min}$ and $\gamma_{\max}$ as in Equation (3.3) if $\mathsf{K}$ is chosen by $\mathsf{AdmSmp}$ above for an arbitrary $\eta \in [n(\lambda)]$ in Theorem 1. We then describe how exactly $\eta$ has to be chosen to yield (balanced) AHFs in Corollary 1.

**Theorem 1** (Adapted from [FHPS13, Jag15])**.** *Let $k, n : \mathbb{N} \to \mathbb{N}$ be polynomially bounded functions with $k(\lambda) \leq n(\lambda)$ for all $\lambda \in \mathbb{N}$, let $Q : \mathbb{N} \to \mathbb{N}$ be a polynomial and let $\varepsilon : \mathbb{N} \to [0,1]$ be a non-negligible function. If $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ is a family of $[n(\ell), k(\ell), n(\ell) \cdot \delta]_2$ error-correcting codes, where $\delta \in (0, 1/2)$ denotes the relative distance of all $C \in \mathcal{C}$, then for $\mathcal{AHF} := \mathcal{C}$ it holds for all $\eta \in [n(\lambda)]$ and $\mathsf{K} \xleftarrow{\$} \{0,1\}_{\perp}^{(n(\lambda),\eta)}$ that*

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

*for*

$$\gamma_{\min}(\lambda) \coloneqq (1 - Q(\lambda) \cdot (1-\delta)^\eta) \cdot 2^{-\eta} \qquad \text{and} \qquad \gamma_{\max}(\lambda) \coloneqq 2^{-\eta} \tag{3.5}$$

*and all $X^{(1)}, \ldots, X^{(Q(\lambda))}, X^* \in \left(\{0,1\}^{k(\lambda)}\right)^{(Q(\lambda)+1)}$ with $X^* \neq X^{(i)}$ for all $i \in [Q]$.*

The proof mostly follows the respective proofs from [FHPS13, Jag15] and we provide it here because Theorem 1 is central to this chapter.

*Proof.* For simplicity, we drop the security parameter throughout the proof. Furthermore, for the remainder of the proof, we fix arbitrary $X^{(1)}, \ldots, X^{(Q)}, X^* \in$

$\{0,1\}^k$ with $X^{(i)} \neq X^*$ for all $i \in [Q]$. First, we analyse the probability for $\mathsf{F}_\mathsf{K}(X^*) = 1$ to hold. Recalling the definition of $\mathsf{F}_\mathsf{K}$ in Equation (3.1), $C(X^*)$ has to be identical with $\mathsf{K}$ on all positions where $\mathsf{K}$ is not $\perp$ for $\mathsf{F}_\mathsf{K}(X^*) = 1$ to hold. Due to the choice of $\mathsf{K}$ as $\mathsf{K} \xleftarrow{\$} \{0,1\}_\perp^{(n,\eta)}$, we have

$$\Pr\left[\mathsf{F}_\mathsf{K}(X^*) = 1\right] = 2^{-\eta},$$

where the probability is taken over the choice of $\mathsf{K}$. This observation already yields the upper bound of $\gamma_{\max}(\lambda) = 2^{-\eta}$. We proceed to upper bound

$$\Pr\left[\mathsf{F}_\mathsf{K}(X^{(i)}) = 1 \mid \mathsf{F}_\mathsf{K}(X^*) = 1\right]$$

for a fixed $i \in [Q]$. Recall that due to the minimal distance of $C$ and because $X^{(i)} \neq X^*$, we have that $C(X^{(i)})$ and $C(X^*)$ have to differ in at least $\delta \cdot n$ positions. Thus, for $\mathsf{F}_\mathsf{K}(X^{(i)}) = \mathsf{F}_\mathsf{K}(X^*) = 1$ to hold, the $\delta \cdot n$ positions in which $C(X^{(i)})$ and $C(X^*)$ differ *have* to coincide with positions where $\mathsf{K}$ is $\perp$. Since we condition on the event $\mathsf{F}_\mathsf{K}(X^*) = 1$, $C(X^*)$ specifies the value of $\mathsf{K}$ in all positions where $\mathsf{K}$ is not $\perp$. Therefore, view the choice of $\mathsf{K}$ as choosing the $\eta$ out of $n$ positions where $\mathsf{K}$ is not $\perp$. Each position has a probability of $(1-\delta)$ for $C(X^{(i)})$ and $C(X^*)$ to not differ. This yields

$$\Pr\left[\mathsf{F}_\mathsf{K}(X^{(i)}) = 1 \mid \mathsf{F}_\mathsf{K}(X^*) = 1\right] \leq (1-\delta)^\eta.$$

Applying the union bound, we then have that

$$\begin{aligned}\Pr\left[\exists\, i \in [Q] : \mathsf{F}_\mathsf{K}(X^{(i)}) = 1 \mid \mathsf{F}_\mathsf{K}(X^*) = 1\right] &\leq \sum_{i=1}^{Q} \Pr\left[\mathsf{F}_\mathsf{K}(X^{(i)}) = 1 \mid \mathsf{F}_\mathsf{K}(X^*) = 1\right] \\ &= Q(1-\delta)^\eta.\end{aligned}$$

We conclude the proof by showing that $\gamma_{\min}$ is a lower bound. We do so by combining the observations we made above:

$$\begin{aligned}&\Pr\left[\mathsf{F}_\mathsf{K}(X^{(1)}) = \cdots = \mathsf{F}_\mathsf{K}(X^{(Q)}) = 0 \wedge \mathsf{F}_\mathsf{K}(X^*) = 1\right] \\ =&\Pr\left[\mathsf{F}_\mathsf{K}(X^{(1)}) = \cdots = \mathsf{F}_\mathsf{K}(X^{(Q)}) = 0 \mid \mathsf{F}_\mathsf{K}(X^*) = 1\right] \cdot \Pr\left[\mathsf{F}_\mathsf{K}(X^*) = 1\right] \\ =&\left(1 - \Pr\left[\exists\, i \in [Q] : \mathsf{F}_\mathsf{K}(X^{(i)}) = 1 | \mathsf{F}_\mathsf{K}(X^*) = 1\right]\right) \cdot \Pr\left[\mathsf{F}_\mathsf{K}(X^*) = 1\right] \\ \geq&(1 - Q(1-\delta)^\eta) \cdot \Pr\left[\mathsf{F}_\mathsf{K}(X^*) = 1\right] \\ =&(1 - Q(1-\delta)^\eta) \cdot 2^{-\eta} = \gamma_{\min}.\end{aligned}$$

□

Having shown lower and upper bounds on the success probability of the partitioning, we now describe how $\eta$ has to be chosen in order to yield a non-negligible success probability for the partitioning using AHFs to succeed. However, our choice of $\eta$ differs from the choice in [Jag15] and further applications of bAHFs such

as [Yam17a, Kat17, Bit20] because we aim at maximizing $\tau$. This will be relevant when we compare VRFs based on bAHFs to VRFs following different paradigms in Section 3.5. In this comparison, we normalize the different constructions to have approximately the same reduction loss. Thus, choosing $\eta$ in a sub-optimal way would put VRFs based on bAHFs in an unfair disadvantage. Therefore, we first show how $\eta$ has to be chosen in order to maximise $\tau(\lambda)$ in Equation (3.4) in Lemma 3 and then show that $\tau(\lambda)$ is non-negligible for this choice of $\eta$ in Corollary 1 and thus (balanced) AHFs can be instantiated from ECCs.

**Lemma 3.** *Let $Q : \mathbb{N} \to \mathbb{N}$ be a polynomial, let $\delta \in (0, 1/2)$, and let $\varepsilon : \mathbb{N} \to (0, 1]$. Then for*

$$\gamma_{\min}(\lambda) = (1 - Q(\lambda) \cdot (1 - \delta)^{\eta}) \cdot 2^{-\eta} \qquad \text{and} \qquad \gamma_{\max}(\lambda) = 2^{-\eta}$$

*from Equation* (3.5) *in Theorem 1 it holds that*

$$\tau(\lambda) = \varepsilon(\lambda)\gamma_{\min}(\lambda) - \frac{\gamma_{\min}(\lambda) - \gamma_{\max}(\lambda)}{2}$$

*from Equation* (3.4) *in Definition 16 is maximal for*

$$\eta = \eta_{\mathsf{opt}} \coloneqq \log_{1-\delta}\left(\frac{-\varepsilon(\lambda) \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot Q(\lambda) \cdot \ln\left(\frac{1-\delta}{2}\right)}\right). \tag{3.6}$$

*Proof.* As in the proof of Theorem 1, we omit the security parameter in the proof for simplicity. Furthermore, since we optimize the choice of $\eta$, we view $\gamma_{\min}, \gamma_{\max}$ and $\tau$ as functions in $\eta$ instead of in $\lambda$ throughout this proof.

In order to compute the maximum of $\tau$, we first need to find the zeros of $\tau'(\eta)$ and then show that $\tau''(\eta)$ is negative for them. Indeed, as we will see, $\eta_{\mathsf{opt}}$ is *the unique* zero of $\tau'(\eta)$. As a preparation, we first compute the first derivatives of $\gamma_{\min}$ and $\gamma_{\max}$. Observe that, we view $\gamma_{\min}$ and $\gamma_{\max}$ as functions of $\eta$ for this proof instead of as functions of $\lambda$. We first claim that

$$\gamma'_{\min}(\eta) = -2^{-\eta}\left(Q \cdot (1 - \delta)^{\eta} \cdot \ln\left(\frac{1-\delta}{2}\right) + \ln(2)\right). \tag{3.7}$$

holds. We prove Equation (3.7) as follows.

$$\begin{aligned}
\gamma'_{\min}(\eta) &= (1 - Q(1-\delta)^{\eta}) \cdot 2^{-\eta})' \\
&= (1 - Q(1-\delta)^{\eta})' \cdot 2^{-\eta} + (1 - Q \cdot (1-\delta)^{\eta}) \cdot (2^{-\eta})' \\
&= -(Q(1-\delta)^{\eta})' \cdot 2^{-\eta} - (1 - Q \cdot (1-\delta)^{\eta}) \cdot \ln(2) \cdot 2^{-\eta} \\
&= 2^{-\eta}(-Q \cdot (1-\delta)^{\eta})' + Q \cdot (1-\delta)^{\eta} \cdot \ln(2) - \ln(2)) \\
&= -2^{-\eta}(Q \cdot \ln(1-\delta) \cdot (1-\delta)^{\eta} - Q \cdot (1-\delta)^{\eta} \cdot \ln(2) + \ln(2)) \\
&= -2^{-\eta}\left(Q \cdot (1-\delta)^{\eta} \cdot \ln\left(\frac{1-\delta}{2}\right) + \ln(2)\right).
\end{aligned}$$

We proceed by observing that

$$\gamma'_{\max}(\eta) = (2^{-\eta})' = -2^{-\eta}\ln(2) \tag{3.8}$$

holds. With these preparations, we claim that

$$\tau'(\eta) = -2^{-\eta} \cdot \underbrace{\left(\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot (1-\delta)^{\eta} \cdot \ln\left(\frac{1-\delta}{2}\right) + \varepsilon \cdot \ln(2)\right)}_{:=\mu(\eta)} \tag{3.9}$$

holds, where we define $\mu(\eta)$ as above to easily reference it later on. We prove Equation (3.9) as follows.

$$\begin{aligned}
\tau'(\eta) &= \varepsilon \cdot \gamma'_{\min}(\eta) - \frac{\gamma'_{\max}(\eta)}{2} + \frac{\gamma'_{\min}(\eta)}{2} \\
&= \left(\varepsilon + \frac{1}{2}\right)\gamma'_{\min}(\eta) - \frac{\gamma'_{\max}(\eta)}{2} \\
&= \left(\varepsilon + \frac{1}{2}\right)\gamma'_{\min}(\eta) + \frac{2^{-\eta}\cdot\ln(2)}{2} && (3.10) \\
&= -\left(\varepsilon + \frac{1}{2}\right)\cdot 2^{-\eta}\cdot\left(Q\cdot(1-\delta)^{\eta}\cdot\ln\left(\frac{1-\delta}{2}\right) + \ln(2)\right) + \frac{2^{-\eta}\cdot\ln(2)}{2} && (3.11) \\
&= -2^{-\eta}\cdot\left(\left(\varepsilon + \frac{1}{2}\right)\cdot Q\cdot(1-\delta)^{\eta}\cdot\ln\left(\frac{1-\delta}{2}\right) + \left(\varepsilon + \frac{1}{2}\right)\ln(2) - \frac{\ln(2)}{2}\right) \\
&= -2^{-\eta}\cdot\left(\left(\varepsilon + \frac{1}{2}\right)\cdot Q\cdot(1-\delta)^{\eta}\cdot\ln\left(\frac{1-\delta}{2}\right) + \varepsilon\cdot\ln(2)\right),
\end{aligned}$$

where Equation (3.10) follows from Equation (3.8) and Equation (3.11) follows from Equation (3.7).

Next, we observe that there is no $\eta \in \mathbb{R}$ such that $2^{-\eta} = 0$. Thus, every zero of $\tau'(\eta)$ has to be a zero of $\mu(\eta)$. Furthermore, we note that every zero of $\mu(\eta)$ is a zero of $\tau'(\eta)$. We proceed by showing that $\eta_{\mathsf{opt}}$ from Equation (3.6) is the only zero of $\mu(\eta)$ and thus also the only zero of $\tau'(\eta)$. It holds that

$$\begin{aligned}
&\mu(\eta) = 0 \\
\Leftrightarrow\ &\left(\varepsilon + \frac{1}{2}\right)\cdot Q\cdot(1-\delta)^{\eta}\cdot\ln\left(\frac{1-\delta}{2}\right) + \varepsilon\cdot\ln(2) = 0 \\
\Leftrightarrow\ &(1-\delta)^{\eta} = \frac{-\varepsilon\cdot\ln(2)}{\left(\varepsilon + \frac{1}{2}\right)\cdot Q\cdot\ln\left(\frac{1-\delta}{2}\right)} \\
\Leftrightarrow\ &\eta = \log_{1-\delta}\left(\frac{-\varepsilon\cdot\ln(2)}{\left(\varepsilon + \frac{1}{2}\right)\cdot Q\cdot\ln\left(\frac{1-\delta}{2}\right)}\right)
\end{aligned}$$

and thus $\eta_{\mathsf{opt}}$ is the only zero of $\mu(\eta)$ and $\tau'(\eta)$. In order to complete the proof it is only left to show that $\tau''(\eta_{\mathsf{opt}}) < 0$ holds. To do so, we first show that

$$\gamma''_{\min}(\eta) = 2^{-\eta}\cdot\left(Q\cdot(1-\delta)^{\eta}\cdot\ln\left(\frac{1-\delta}{2}\right)\cdot\ln\left(\frac{2}{1-\delta}\right) + \ln(2)^2\right) \tag{3.12}$$

holds. We prove Equation (3.12) as follows:

$$\begin{aligned}
\gamma''_{\min}(\eta) &= ((1 - Q(1-\delta)^\eta) \cdot 2^{-\eta})'' \\
&= \left(-2^{-\eta}\left(Q \cdot (1-\delta)^\eta \cdot \ln\left(\frac{1-\delta}{2}\right) + \ln(2)\right)\right)' \qquad (3.13) \\
&= \left(-2^{-\eta} \cdot Q \cdot (1-\delta)^\eta \cdot \ln\left(\frac{1-\delta}{2}\right)\right)' - (2^{-\eta} \cdot \ln(2))' \\
&= Q \cdot \ln\left(\frac{1-\delta}{2}\right)(-2^{-\eta} \cdot (1-\delta)^\eta)' + 2^{-\eta}\ln(2)^2 \\
&= Q \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \Big((-2^{-\eta})' \cdot (1-\delta)^\eta \ -2^{-\eta} \cdot ((1-\delta)^\eta)'\Big) + 2^{-\eta}\ln(2)^2 \\
&= Q \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \Big(\ln(2) \cdot 2^{-\eta} \cdot (1-\delta)^\eta \\
&\qquad - \Big(2^{-\eta} \cdot \ln(1-\delta) \cdot (1-\delta)^\eta\Big)\Big) + 2^{-\eta} \cdot \ln(2)^2 \\
&= Q \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot (\ln(2) - \ln(1-\delta))\,(1-\delta)^\eta 2^{-\eta} + 2^{-\eta} \cdot \ln(2)^2 \\
&= Q \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right)(1-\delta)^\eta 2^{-\eta} + 2^{-\eta} \cdot \ln(2)^2 \\
&= 2^{-\eta} \cdot \left(Q \cdot (1-\delta)^\eta \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right) + \ln(2)^2\right)
\end{aligned}$$

where Equation (3.13) follows from Equation (3.7). Similar to before, we observe that

$$\gamma''_{\max}(\eta) = \Big(-2^{-\eta} \cdot \ln(2)\Big)' = 2^{-\eta} \cdot \ln(2)^2 \qquad (3.14)$$

holds. With these preparations at hand, we claim that

$$\tau''(\eta) = 2^{-\eta} \cdot \underbrace{\left(\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot (1-\delta)^\eta \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right) + \varepsilon \cdot \ln(2)^2\right)}_{:=\nu(\eta)} \qquad (3.15)$$

holds, where we define $\nu(\eta)$ as above in order to easily reference it later on. We

prove Equation (3.15) as follows:

$$\begin{aligned}
\tau''(\eta) &= \left(\varepsilon + \frac{1}{2}\right) \cdot \gamma''_{\min}(\eta) - \frac{\gamma''_{\max}(\eta)}{2} \\
&= \left(\varepsilon + \frac{1}{2}\right) \cdot \gamma''_{\min}(\eta) - \frac{2^{-\eta} \cdot \ln(2)^2}{2} \\
&= 2^{-\eta} \cdot \left(\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot (1-\delta)^{\eta} \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right) \right. \\
&\qquad \left. + \left(\varepsilon + \frac{1}{2}\right) \ln(2)^2 - \frac{\ln(2)^2}{2}\right) \\
&= 2^{-\eta} \cdot \left(\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot (1-\delta)^{\eta} \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right) + \varepsilon \ln(2)^2\right)
\end{aligned}$$

We observe that $2^{-\eta}$ is positive for all $\eta \in \mathbb{R}$ and thus $\tau''(\eta_{\mathsf{opt}}) < 0$ holds if and only if $\nu(\eta_{\mathsf{opt}}) < 0$. As we show below, this is the case. It holds that

$$\begin{aligned}
\nu(\eta_{\mathsf{opt}}) &= \left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot (1-\delta)^{\eta_{\mathsf{opt}}} \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right) + \varepsilon \cdot \ln(2)^2 \\
&= \left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot \frac{-\varepsilon \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot \ln\left(\frac{1-\delta}{2}\right)} \cdot \ln\left(\frac{1-\delta}{2}\right) \cdot \ln\left(\frac{2}{1-\delta}\right) + \varepsilon \cdot \ln(2)^2 \\
&= -\varepsilon \ln(2) \cdot \ln\left(\frac{2}{1-\delta}\right) + \varepsilon \cdot \ln(2)^2 \\
&= \varepsilon \cdot \ln(2) \cdot (\ln(2) - (\ln(2) - \ln(1-\delta))) \\
&= \underbrace{\varepsilon \cdot \ln(2)}_{>0} \cdot \underbrace{\ln(1-\delta)}_{<0} < 0, \qquad (3.16)
\end{aligned}$$

where the inequality in Equation (3.16) holds since $\varepsilon \cdot \ln(2)$ is always positive and $\ln(1-\delta)$ is negative because $\delta \in (0, 1/2)$. This concludes the proof of Lemma 3. □

Now that we have shown how to choose $\eta$ in order to maximize $\tau$, we still have to show that this ideal choice indeed leads to a non-negligible security loss, which we do in the following corollary. We further observe that $\eta_{\mathsf{opt}}$ is not always an integer. Thus, we round it up to the next largest integer to use it in $\mathsf{bAdmSmp}$.

**Corollary 1.** *Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of $[n(\ell), k(\ell), n(\ell) \cdot \delta]_2$ error-correcting codes, where $k$ and $n$ are polynomially bounded and $\delta \in (0, 1/2)$ denotes the common relative distance of all $C \in \mathcal{C}$. Furthermore, for all polynomials $Q : \mathbb{N} \to \mathbb{N}$ and a non-negligible function $\varepsilon : \mathbb{N} \to [0, 1]$ let $\mathsf{bAdmSmp}(1^\lambda, Q(\lambda), \varepsilon(\lambda))$ be the algorithm that sets*

$$\eta := \left\lceil \log_{1-\delta}\left(\frac{-\varepsilon(\lambda) \cdot \ln(2)}{\left(\varepsilon(\lambda) + \frac{1}{2}\right) \cdot Q(\lambda) \cdot \ln\left(\frac{1-\delta}{2}\right)}\right)\right\rceil$$

*and outputs* $\mathsf{K} \xleftarrow{\$} \{0,1\}_{\perp}^{n(\lambda),\eta}$. *Then for* $\gamma_{\min}$ *and* $\gamma_{\max}$ *from Theorem 1 and* $\tau$ *from Equation* (3.4)*,* $\gamma_{\min}$ *and* $\tau$ *are both non-negligible. In particular,* $\mathcal{AHF} := \mathcal{C}$ *is a family of balanced admissible hash functions and if* $\varepsilon(\lambda)$ *is set to a positive constant it immediately follows that* $\mathcal{AHF}$ *is also a family of admissible hash functions.*

*Remark* 5. Note that the choices of $\eta$ above, just as the choices in literature such as those in [FHPS13, Jag15], can yield a $\eta$ larger than $n$ for small security parameters. However, $\eta \in [n]$ holds for reasonable parameters as presented in Section 3.5.

*Proof.* Analogous to the proof of Theorem 1, we drop the security parameter for simplicity. First observe that we can simplify the term $\tau$ as follows for all $\eta \in [n]$:

$$\begin{aligned}
\tau &= \varepsilon \cdot \gamma_{\min} - \frac{\gamma_{\max} - \gamma_{\min}}{2} \\
&= \varepsilon \cdot (1 - Q(1-\delta)^{\eta}) \cdot 2^{-\eta} - \frac{2^{-\eta} - (1 - Q(1-\delta)^{\eta}) \cdot 2^{-\eta}}{2} \\
&= 2^{-\eta} \left( \varepsilon \cdot (1 - Q(1-\delta)^{\eta}) - \frac{1 - (1 - Q(1-\delta)^{\eta})}{2} \right) \\
&= 2^{-\eta} \left( \varepsilon \cdot (1 - Q(1-\delta)^{\eta}) - \frac{Q(1-\delta)^{\eta}}{2} \right) \\
&= 2^{-\eta} \left( \varepsilon - Q(1-\delta)^{\eta} \left( \varepsilon + \frac{1}{2} \right) \right) \qquad (3.17)
\end{aligned}$$

Next, we simplify the term $Q(1-\delta)^{\eta}$ for $\eta$ as in Corollary 1 above.

$$\begin{aligned}
Q(1-\delta)^{\eta} &= Q(1-\delta)^{\left\lceil \log_{1-\delta}\left( \frac{-\varepsilon \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot \ln\left(\frac{1-\delta}{2}\right)} \right) \right\rceil} \\
&\leq Q(1-\delta)^{\log_{1-\delta}\left( \frac{-\varepsilon \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot \ln\left(\frac{1-\delta}{2}\right)} \right)} \\
&= \frac{-Q \cdot \varepsilon \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot Q \cdot \ln\left(\frac{1-\delta}{2}\right)} \\
&= \frac{-\varepsilon \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot \ln\left(\frac{1-\delta}{2}\right)} \qquad (3.18)
\end{aligned}$$

For simplicity, we substitute the non-negligible term from Equation (3.18) by $\alpha$. That is, we set

$$\alpha := \frac{-\varepsilon \cdot \ln(2)}{\left(\varepsilon + \frac{1}{2}\right) \cdot \ln\left(\frac{1-\delta}{2}\right)}.$$

We observe that $\alpha$ is non-negligible because $\varepsilon$ is non-negligible and $\delta$ is constant. We now proceed by simplifying the term $2^{-\eta}$. For this, we use the fact that for all

$r \in \mathbb{R}^{>0}$ it holds that $\log_{1-\delta}(r) = \log(r)/\log(1-\delta)$. We thus have the following:

$$\begin{aligned}
2^{-\eta} &= 2^{-\left\lceil \log_{1-\delta}\left(\frac{-\varepsilon\cdot\ln(2)}{\left(\varepsilon+\frac{1}{2}\right)\cdot Q\cdot\ln\left(\frac{1-\delta}{2}\right)}\right)\right\rceil} \\
&\geq 2^{-\log_{1-\delta}\left(\frac{-\varepsilon\cdot\ln(2)}{\left(\varepsilon+\frac{1}{2}\right)\cdot Q\cdot\ln\left(\frac{1-\delta}{2}\right)}\right)-1} \\
&= \frac{1}{2}\cdot\left(\frac{-\varepsilon\cdot\ln(2)}{\left(\varepsilon+\frac{1}{2}\right)\cdot Q\cdot\ln\left(\frac{1-\delta}{2}\right)}\right)^{-\frac{1}{\log(1-\delta)}} \\
&= \frac{1}{2}\cdot Q^{\frac{1}{\log(1-\delta)}}\cdot\alpha^{-\frac{1}{\log(1-\delta)}} \qquad (3.19)
\end{aligned}$$

Combining Equation (3.17), Equation (3.18) and Equation (3.19), we conclude that

$$\begin{aligned}
\tau &= 2^{-\eta}\left(\varepsilon - Q(1-\delta)^{\eta}\left(\varepsilon+\frac{1}{2}\right)\right) \\
&\geq \frac{1}{2}\cdot Q^{\frac{1}{\log(1-\delta)}}\alpha^{-\frac{1}{\log(1-\delta)}}\left(\varepsilon-\alpha\left(\varepsilon+\frac{1}{2}\right)\right).
\end{aligned}$$

Finally, using that $1/\log(1-\delta) < 0$ because $\delta \in (0, 1/2)$, $\alpha$ is non-negligible and that $Q$ is a polynomial, we conclude that $\tau$ is non-negligible. Analogously, we conclude that

$$\gamma_{\min} = (1 - Q(1-\delta)^{\eta})\cdot 2^{-\eta} \geq \frac{1}{2}\cdot(1-\alpha)Q^{\frac{1}{\log(1-\delta)}}\cdot\alpha^{-\frac{1}{\log(1-\delta)}}$$

is non-negligible because $\alpha$ is non-negligible, $\delta \in (0, 1/2)$ is a constant and $Q$ is a polynomial. □

### 3.2.3 Efficiency Bounds for Admissible Hash Functions from Coding Theory

When AHFs are used in cryptographic schemes, the length $n$ of code words of the used ECC usually affects the size of private and secret keys, signatures and proofs. For example, the public key of the VRF of [Jag15] contains two groups element for every bit in the output of the ECC. The VRFs in [Kat17, Yam17a] reduce this, but still contain at least logarithmically (in $n$) many group elements. Even though asymptotically logarithmic, the number may still be impractically large when instantiated concretely - we explore this in detail in Section 3.5. Furthermore, as we have seen in Corollary 1, the relative distance $\delta$ of the ECC affects the probability that the partitioning succeeds.

Thus, in order to be able to efficiently instantiate AHFs, it is important to reduce the length $n$ of code words while at the same time maintaining a high relative distance $\delta$. In the following, we analyse the inherent limitations of instantiating AHFs with binary ECCs, by applying upper and lower bounds from coding theory.

The Gilbert-Varshamov bound. In order to instantiate AHFs efficiently, we need a code that has both a high rate and a high relative minimal distance. Coding theorists worked on the construction of such codes and accompanying bounds for decades [MS98, HP03]. Asymptotically, the *Gilbert-Varshamov* (GV) bound guarantees the *existence* of well-suited families of binary ECCs, but we note that this result is *not constructive*.

**Theorem 2** (Gilbert-Varshamov bound [Gil52, Var57])**.** *For all $n \in \mathbb{N}$ and $c \in (0, 1/2)$, there exists an ECC $C : \mathbb{F}_2^k \to \mathbb{F}_2^n$ with*

$$\delta(C) \geq c \qquad \text{and} \qquad \mathcal{R}(C) \geq 1 - H_2(\delta(C)),$$

*where $H_2$ denotes the binary entropy, defined as*

$$H_2(p) := p \cdot \log(1/p) + (1-p) \cdot \log(1/(1-p)).$$

*for all $p \in (0, 1)$.*

Even though the GV bound guarantees the existence of families of binary ECCs with the parameters from above and random linear codes attain the bound [Var57], no explicit construction of a family of ECCs attaining the GV bound is known so far. Unfortunately, rejection sampling of random linear codes in order to find one with a large minimal distance also is infeasible, because, as [Var97, DMS99] show, there is no efficient algorithm that can compute or approximate the minimal distance of a random linear code. It is possible however to construct a family of binary ECCs that comes relatively close to this bound by concatenating algebraic geometry codes with binary error-correcting codes [SAK+01, Section V].

Hence, when instantiating a family of (balanced) AHFs, we can treat the GV bound as an optimistic upper bound on what is possible with currently known families of binary ECCs. For example the GV bound yields that for $\delta = 0.2$, the best rate we can hope to achieve with known construction of families of ECCs is $\approx 0.28$. For the VRF from [Jag15], this would mean that the required number of group elements in the public key is at least about four times larger than the number of input bits. We show in Section 3.5 that similar numbers apply to more recent VRFs.

Efficiency bounds for concrete security parameters. Albeit the unpromising situation in the asymptotic setting, it is worth investigating the situation for specific practically relevant security parameters and input lengths $k$ of ECCs[1] instead of only considering whole families of ECCs. That is, for example for $k = 128$, there is a $[255, 128, 38]_2$ ECC $C$ based on a BCH code [Gra07]. Thus, this code beats the GV bound with $\mathcal{R}(C) \approx 1/2$ and $\delta(C) \approx 0.15$, whereas the GV bound only guarantees the existence of a ECC $C$ with $\mathcal{R}(C) \approx 0.39$ and $\delta(C) = 0.15$. To asses this concrete setting more thoroughly, we consider general upper bounds

[1] For the VRFs in [HJ16, Jag15, Kat17, Koh19, Yam17a] this is identical to the input length.

on the relation between the rate and the relative minimal distance. The sharpest such bound for binary ECCs, to the best of our knowledge, is the MRRW bound presented in Theorem 3.

**Theorem 3** (MRRW bound [MRRW77])**.** *Let $C$ be an $[n, k, d]_2$ ECC with relative distance $\delta(C) \in (0, 1/2)$ and let $g(x) := H_2((1 - \sqrt{1-x})/2)$. Then*

$$\mathcal{R}(C) \leq \min_{0 \leq u \leq 1-2\,\delta} \{1 + g(u^2) - g(u^2 + 2\,\delta(C)u + 2\,\delta(C))\}$$

*holds for the rate of $C$.*

The MRRW-Bound thus also yields limits on what can be achieved for specific security parameters instead of asymptotically for all security parameters. For example, every binary ECC $C$ with $\delta(C) = 0.15$ inevitably has $\mathcal{R}(C) < 0.58$. Analogously, every ECC $C$ with $\delta(C) = 0.2$ has $\mathcal{R}(C) < 0.47$.

The GV bound and the MRRW bound above enable us to realistically asses the size of public keys, private keys and proofs of VRFs that are based on AHFs. However, the bounds already indicate that instantiations of VRFs based on AHFs will not be very efficient. Thus, we will first consider VRFs that use computational assumptions as replacements for information-theoretic AHFs in Section 3.3 and Section 3.4 before we compare the efficiency of different VRFs in Section 3.5 by applying the GV bound and the MRRW bound.

## 3.3 Verifiable Random Functions from Computational Admissible Hash Functions

In this section, we will show how the inherent limitations of (balanced) AHFs due to their instantiation with ECCs can be overcome. We do so by *relaxing the constraints* and consider a *computational* setting and replace ECCs with cryptographic hash functions. Throughout this section we will first introduce *computational admissible hash function*s (cAHFs) as a drop-in replacement for bAHFs with computational security. We then introduce *truncation-collision resistant* (TCR) hash functions from [JK18] and show how they can be used to instantiate cAHFs. Finally, we demonstrate how cAHFs can be used as a drop-in replacement for bAHFs by presenting a variant of Jager's VRF that uses cAHFs instead of bAHFs.

High-level perspective. In order to sketch the main idea, consider VRFs as an example. Our approach is to let the reduction guess the first $\eta = \mathcal{O}(\log \lambda)$ bits of $H(X^*)$, where $H \xleftarrow{\$} \mathcal{H}$ is a cryptographic hash function and $X^*$ is the challenge input chosen by the adversary. Using standard techniques from selectively-secure constructions, we prove security with a reduction that is successful if the first $\eta$ bits of $H(X^*)$ are guessed correctly, while the hash of every input for that the adversary requests the evaluation of the VRF differs in at least one of the first $\eta$ bits. For this

approach to yield a reduction with non-negligible loss, we have to choose $\eta$ such that it fulfills the following two conflicting goals:

1. $\eta$ has to be small enough, such that the probability of guessing the first $\eta$ bits of $H(X^*)$ correctly is non-negligible

2. $\eta$ has to be large enough to ensure that it is unlikely, relative to the adversary's advantage, to make a query $X$ whose hash also matches on the first $\eta$ guessed bits ("truncation-collision resistance").

Following [JK18] (which in turn is inspired by [BHJ+13, BHJ+15]), we balance these two goals by choosing $n'$ depending on the runtime and advantage of the adversary.

This relaxation to computational instead of information-theoretic security frees us from the redundancy inherent to ECCs. As we show in Section 3.5, this allows us to construct VRFs with significantly smaller public and secret keys, and proofs. Furthermore, we aim at defining cAHFs in a way that allows us to replace bAHFs in many constructions like [AMN+19, HJ16, Jag15, Kat17, LST18, Yam17a], while making only minor modifications to the constructions or their accompanying security proofs.

Computational admissible hash functions. We keep using the function $\mathsf{F}$, since we want to design cAHFs such that the can *generically replace* bAHFs. Allowing $H$ to have pairs of inputs $X, Y$ with $X \neq Y$ and $H(X) = H(Y)$ comes with the problem that an adversary can have a collision for $H$ hard coded. Therefore, we need to draw the function $H$ from a *family of hash functions* $\mathcal{H}$. This requires us to make the following minor modification to the definition of $\mathsf{F}$ of replacing the security parameter index by a hash function $H$ from a family of hash functions $\mathcal{H}$. Note that we consider a single family of functions $\mathcal{H}$ and not a family of hash functions for each potential $\lambda$ to better capture a real-world instantiation with already standardized hash functions.

$$\mathsf{F}_{\mathsf{K},H}(X) = \begin{cases} 1, & \text{if } \forall\, j \in [n] : H(X)_j = \mathsf{K}_j \vee \mathsf{K}_j = \bot \\ 0, & \text{otherwise.} \end{cases} \tag{3.20}$$

After these adaptations to the computational setting, we are ready to define computational admissible hash functions.

**Definition 19** (*Computational admissible hash function*s (cAHFs))**.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions for some $n \in \mathbb{N}$ and let $H \in \mathcal{H}$. For all $\mathsf{K} \in \{0,1,\bot\}^n$ and all sequences $(X^{(1)}, \ldots, X^{(Q)}, X^*)$ with $X^{(i)}, X^* \in \{0,1\}^*$ and $X^{(i)} \neq X^*$ for all $i$, we let $X^{(Q+1)} := X^*$ to ease notation and define the events

coll, badChal and badEval as follows.

$$\begin{aligned}
\mathsf{badChal} &\iff \mathsf{F}_{\mathsf{K},H}(X^*) = 0 \\
\mathsf{coll} &\iff \exists\, i, j \text{ with } X^{(i)} \neq X^{(j)} \text{ s.t.} \\
&\qquad \forall\, \ell \in [n] : H(X^{(i)})_\ell = H(X^{(j)})_\ell \vee \mathsf{K}_\ell = \perp \\
\mathsf{badEval} &\iff \exists\, i \in [Q] \text{ s.t. } \mathsf{F}_{\mathsf{K},H}(X^{(i)}) = 1
\end{aligned}$$

Let $t_{\mathcal{A}} \in \mathbb{N}$ and $\varepsilon_{\mathcal{A}} \in (0,1]$ such that $t_{\mathcal{A}}/\varepsilon_{\mathcal{A}} < 2^\lambda$, where $\lambda$ is the security parameter. We say that $\mathcal{H}$ is a family of *computational admissible hash function*s (cAHFs), if there is an efficient algorithm $\mathsf{cAdmSmp}(1^\lambda, t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ generating $\mathsf{K} \in \{0,1,\perp\}^n$ such that for every adversary $\mathcal{A}$ running in time $t_{\mathcal{A}}$ outputting $(X^{(1)}, \ldots, X^{(Q)}, X^*)$ it holds that

$$\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal},$$

badChal and coll are independent, and that

$$\tau(\lambda) \coloneqq \Pr[\neg\mathsf{badChal}](\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]) \tag{3.21}$$

is non-negligible as a function in $\lambda$. The probabilities are over the randomness used by $\mathcal{A}$, $H \xleftarrow{\$} \mathcal{H}$ and $\mathsf{K} \xleftarrow{\$} \mathsf{cAdmSmp}(1^\lambda, t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$.

*Remark* 6. The term $\tau$ in Equation (3.21) is the equivalent of $\tau$ for bAHFs in Definition 16, in the sense that it conveniently describes a term that typically occurs in a reduction-based security proof that uses a cAHF. Intuitively, it captures a security proof that in a first step aborts when coll occurs and in a second step aborts when badChal occurs. We will study a concrete application in Section 3.3.2.

### 3.3.1 Computational Admissible Hash Functions from Truncation Collision Resistance

We show how to construct very efficient cAHFs based on *truncation-collision resistant* (TCR) hash functions, as introduced in [JK18].

Truncation collision resistant hash functions. For a hash function $H : \{0,1\}^* \to \{0,1\}^n$ we write $H_{:\eta} : \{0,1\}^* \to \{0,1\}^\eta$ to denote the hash function $H$, with outputs truncated to the first $\eta \in [n]$ bits. Essentially, a hash function is *truncation collision resistant*, if for every prefix of length $\eta \in [n]$ there is no significantly more efficient algorithm to find a collision for $H_{:\eta}$ than the birthday attack. Note that this property is likely satisfied by standard cryptographic hash functions, like SHA-3.

**Definition 20** (Truncation collision resistance [JK18])**.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions. For $\eta \in [n]$, we say that an adversary $\mathcal{A}$ $\eta$-breaks the truncation collision resistance of $\mathcal{H}$, if it runs in time $t_{\mathcal{A}}$ and

$$\Pr_{H \xleftarrow{\$} \mathcal{H}}\left[\begin{array}{l}(x_0, \ldots x_q) \xleftarrow{\$} \mathcal{A}(H) : \\ \exists\, u, v \text{ s.t. } H_{:\eta}(x_u) = H_{:\eta}(x_v) \wedge x_u \neq x_v\end{array}\right] > \frac{t_{\mathcal{A}}(t_{\mathcal{A}}-1)}{2^{\eta+1}}.$$

We say $\mathcal{H}$ is *truncation collision resistant*, if there exists no adversary $\mathcal{A}$ that $\eta$-breaks the truncation collision resistance of $H$ for any $\eta \in [n]$.

We deem truncation collision resistance a reasonable assumption since truncated versions of SHA-256 (to 224 bits) and SHA-512 (to 384 bits) have already been standardized by NIST in [oST15a]. Furthermore, [oST15b] defines *extendable-output functions* (XOF) based on SHA-3. These allow to extend the output of the hash function to an arbitrary length while maintaining collision resistance.

Useful technical lemma. The following lemma is a variant of Lemma 1 by Jager and Kurek in [JK18], tailored to our application and cAHFs, which yields better parameters than the corresponding result in [JK18]. The condition $t/\varepsilon < 2^\lambda$ captures that we consider an efficient adversary, *i.e.* one with a small time complexity, a high success probability or both. Essentially, the lemma guarantees that if we choose $\eta \coloneqq \lceil \log(4t(2t-1)/\varepsilon) \rceil$ then the adversary is sufficiently unlikely to find a collision on $H_{:\eta}$ if $H$ is a TCR hash function. Furthermore it guarantees that the size of $\{0,1\}^\eta$ is polynomial and a reduction can thus guess a value in this space correctly with a non-negligible probability. Finally, it also guarantees that $\eta$ is at most $2\lambda + 3$.

**Lemma 4.** *Let $t \in \mathbb{N}$, $\varepsilon \in (0,1]$ such that $t/\varepsilon < 2^\lambda$, and $\eta \coloneqq \lceil \log(4t(2t-1)/\varepsilon) \rceil$. Then it holds that*

$$\eta \in \{1, \ldots, 2\lambda + 3\}, \qquad \frac{2t(2t-1)}{2^\eta} \leq \frac{\varepsilon}{2} \qquad \text{and} \qquad \frac{1}{2^\eta} \geq \frac{\varepsilon}{16t^2 - 8t}$$

*Proof.* We start by proving $\eta \in \{1, \ldots, 2\lambda + 3\}$.

$$\begin{aligned} \eta = \lceil \log(4t(2t-1)/\varepsilon) \rceil &\leq \left\lceil \log\left(4 \cdot 2^\lambda (2t-1)\right) \right\rceil \\ &\leq \left\lceil \log\left(8 \cdot 2^\lambda t\right) \right\rceil \leq \left\lceil \log\left(2^\lambda 2^{\lambda+3}\right) \right\rceil = 2\lambda + 3 \end{aligned}$$

Since $4t(2t-1) = 8t^2 - 4t > 1$ for all $t \in \mathbb{N}$ and $\varepsilon \in (0,1]$, we have $\log(4t(2t-1)/\varepsilon) > 0$ and therefore $\eta \geq 1$.

We proceed to prove $2t(2t-1)/2^\eta \leq \varepsilon/2$.

$$\frac{2t(2t-1)}{2^\eta} = \frac{2t(2t-1)}{2^{\lceil \log(4t(2t-1)/\varepsilon) \rceil}} \leq \frac{\varepsilon 2t(2t-1)}{4t(2t-1)} = \frac{\varepsilon}{2}$$

Finally, we have

$$\frac{1}{2^\eta} = \frac{1}{2^{\lceil \log(4t(2t-1)/\varepsilon) \rceil}} \geq \frac{1}{2} \cdot \frac{\varepsilon}{4t(2t-1)} = \frac{\varepsilon}{16t^2 - 8t}.$$

□

Constructing cAHFs from TCRs. Lemma 4 allows us to prove that a family of TCRs is also a family of cAHFs. Note that even though the first definition of AHFs in [BB04b] already incorporates collision resistant hash functions – they are only used to enable the processing of arbitrary length inputs – while the core of the AHF in [BB04b] is the error correcting code that yields an information-theoretic AHF, which we replace with TCRs hash functions.

**Theorem 4.** *Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^{2\lambda+3}\}$ be a family of truncation collision resistant hash functions in the sense of Definition 20. Then $\mathcal{H}$ is a family of cAHFs. In particular, let $t_{\mathcal{A}} \in \mathbb{N}$ and $\varepsilon_{\mathcal{A}} \in (0,1]$ such that $t_{\mathcal{A}}/\varepsilon_{\mathcal{A}} < 2^{\lambda}$. Then there is an algorithm $\mathsf{cAdmSmp}(1^{\lambda}, t, \varepsilon)$ such that for every adversary $\mathcal{A}$ running in time $t_{\mathcal{A}}$ that, given $H \xleftarrow{\$} \mathcal{H}$, outputs $X^{(1)}, \ldots, X^{(Q)}, X^* \in \{0,1\}^*$ with $X^{(i)} \neq X^*$ it holds that* coll *and* badChal *are independent,*

$$\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal}$$

*and*

$$\Pr[\neg\mathsf{badChal}](\varepsilon - \Pr[\mathsf{coll}]) \geq \varepsilon^2/(32t^2 - 16t).$$

*In particular, if $t_{\mathcal{A}}$ is polynomial in $\lambda$ and $\varepsilon$ is non-negligible in $\lambda$, then $\varepsilon_{\mathcal{A}}^2/(32t_{\mathcal{A}}^2)$ is also non-negligible and $\mathcal{H}$ is therefore a family of cAHFs.*

*Proof.* Let $n := 2\lambda+3$. The algorithm $\mathsf{cAdmSmp}(1^{\lambda}, t, \varepsilon)$ sets $\eta := \lceil \log(4t(2t-1)/\varepsilon) \rceil$, samples $\mathsf{K}' \xleftarrow{\$} \{0,1\}^{\eta}$, and defines $\mathsf{K} := K' \,\|\, \perp^{n-\eta}$, where $\|$ denotes string concatenation and $\perp^{n-\eta}$ the string consisting of $(n-\eta)$-times the $\perp$-symbol. In total the key $\mathsf{K}$ consists of $\eta$ uniformly random bits, padded to a string of length $n$ in $\{0,1,\perp\}^n$ by appending $\perp$-symbols. Note that $n \geq \eta$ by Lemma 4.

Recall that $\neg$badChal occurs *if and only if* $\mathsf{F}_{\mathsf{K},H}(X^*) = 1$. For our construction, this means that the first $\eta$ bits of $H(X^*)$ are *identical* to $\mathsf{K}'$, the first $\eta$ bits of $\mathsf{K}$. Since $\mathsf{K}'$ is chosen uniformly at random, and independent of $H \xleftarrow{\$} \mathcal{H}$, this happens with probability $2^{-\eta}$, and therefore

$$\Pr[\neg\mathsf{badChal}] = \frac{1}{2^{\eta}} \geq \frac{\varepsilon_{\mathcal{A}}}{16t_{\mathcal{A}}^2 - 8t},$$

where the inequality uses Lemma 4. Furthermore, recall that coll occurs, if the adversary outputs, as queries or as challenge, two values $X \neq Y$ such that $H(X)$ and $H(Y)$ are identical in all positions where $\mathsf{K}$ is not $\perp$. In particular, we then have $H_{:\eta}(X) = H_{:\eta}(Y)$. Therefore, we claim that

$$\Pr[\mathsf{coll}] \leq \frac{\varepsilon_{\mathcal{A}}}{2}.$$

We prove this upper bound on Pr[coll] by contradiction. Assume $\mathcal{A}$ outputs $X = (X^{(1)}, \ldots, X^{(Q)}, X^*)$ such that $\Pr[\mathsf{coll}] > \varepsilon/2$. Then we can construct an adversary $\mathcal{B}$ that $\eta$-breaks the truncation collision resistance of $\mathcal{H}$. $\mathcal{B}$ runs $\mathcal{A}$, waits for $\mathcal{A}$ to output $X$ and then outputs $X$ itself. $\mathcal{B}$'s running time consists of the time to execute

$\mathcal{A}$ plus the time to output $X$, yielding[2] $t_{\mathcal{B}} \leq 2 \cdot t_{\mathcal{A}}$. Thus, $\mathcal{B}$ is an algorithm with success probability at least $\varepsilon_{\mathcal{A}}/2$ in $\eta$-breaking the truncation collision resistance of $\mathcal{H}$. We furthermore have

$$\Pr[\mathsf{coll}] > \varepsilon_{\mathcal{A}}/2 \geq \frac{2t_{\mathcal{A}}(2t_{\mathcal{A}}-1)}{2^{\eta}} \geq \frac{2t_{\mathcal{A}}(2t_{\mathcal{A}}-1)}{2^{\eta}} = \frac{t_{\mathcal{B}}(t_{\mathcal{B}}-1)}{2^{\eta}} > \frac{t_{\mathcal{B}}(t_{\mathcal{B}}-1)}{2^{\eta+1}},$$

where the second inequality follows from Lemma 4. This contradicts the truncation collision resistance of $\mathcal{H}$ and therefore proves the upper bound on $\Pr[\mathsf{coll}]$. We thus conclude that

$$\Pr[\neg\mathsf{badChal}](\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]) \geq \frac{\varepsilon_{\mathcal{A}}}{16t_{\mathcal{A}}^2 - 8t}\left(\varepsilon_{\mathcal{A}} - \frac{\varepsilon_{\mathcal{A}}}{2}\right) = \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t}$$

holds. Note that if $t_{\mathcal{A}}$ is polynomial in $\lambda$ and $\varepsilon_{\mathcal{A}}$ is non-negligible in $\lambda$, then $\varepsilon_{\mathcal{A}}^2/(32t_{\mathcal{A}}^2 - 16t)$ is also non-negligible.

Moreover, the events $\mathsf{coll}$ and $\mathsf{badChal}$ are independent of each other because $\mathsf{K}$ is chosen independently from $(X^{(1)}, \ldots, X^{(Q)}, X^*)$. Finally, we explain that if $\mathsf{badEval}$ occurs, then either $\mathsf{badChal}$ or $\mathsf{coll}$ *must occur*. This is because if there exists $x \in \{X^{(1)}, \ldots, X^{(Q)}\}$ with $x \neq X^*$ and $H_{:\eta}(x) = \mathsf{K}'$, then we have either that also $H_{:\eta}(X^*) = \mathsf{K}'$ and thus $\mathsf{coll}$ occurs, or we have that there exists an index $i \in [\eta]$ such that $H(X^*)_i \neq \mathsf{K}'_i$ and thus $\mathsf{badChal}$ occurs. This concludes the proof. □

### 3.3.2 Verifiable Random Functions from Computational Admissible Hash Functions

Now that we have constructed cAHFs from TCRs hash functions, we demonstrate how they can be used in a construction as a drop-in replacement for bAHFs. We do so using Jager's VRF [Jag15] as an example. We then show how cAHFs are used in a proof in Section 3.3.2. Furthermore, we improve Jager's VRF with a new proof technique that allows us to halve the size of public verification keys and secret keys.

**Construction 1.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions for some $n \in \mathbb{N}$, let $\mathsf{GrpGen}$ be a certified bilinear group generator (see Definition 10) and let $\mathcal{VRF}_{\mathsf{cAHF}} = (\mathsf{SetupVRF}_{\mathsf{cAHF}}, \mathsf{Eval}_{\mathsf{cAHF}}, \mathsf{Vfy}_{\mathsf{cAHF}})$ be the following algorithms.

**Key generation:** $\mathsf{SetupVRF}_{\mathsf{cAHF}}(1^\lambda)$ runs $\mathcal{BG} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$, chooses a random hash function $H \xleftarrow{\$} \mathcal{H}$ and random generators $g, h \xleftarrow{\$} \mathbb{G}^*$, where $\mathbb{G}$ is from $\mathcal{BG}$, and $w_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [n+1]_0$. It then defines $g_i := g^{w_i}$ and the keys as

$$\mathsf{vk} := \left(H, \mathcal{BG}, g, h, (g_i)_{i\in[n+1]_0}\right) \qquad \text{and} \qquad \mathsf{sk} := (w_i)_{i\in[n+1]_0}.$$

[2] One could tighten the upper bound $t_{\mathcal{B}}$ to $t_{\mathcal{A}} + Q$. However, it would at most save a factor of two in the run time of $\mathcal{B}$ and would complicate the analysis. Therefore, we use the slightly less tight bound.

**Evaluation:** On input $\mathsf{vk}, \mathsf{sk}$ and $X \in \{0,1\}^*$, $\mathsf{Eval}_{\mathsf{cAHF}}$ computes

$$w_X \coloneqq w_0 \cdot \left( \prod_{i=1}^{n} w_i^{H(X)_i} \right) \cdot w_{n+1} \qquad \text{and} \qquad Y \coloneqq e(g,h)^{w_X}.$$

To compute the proof, it sets $\pi_0 \coloneqq g^{w_0}$ and then computes $\pi_1, \ldots, \pi_n$ as

$$\pi_i \coloneqq \pi_{i-1}^{\binom{w_i^{H_i}}{}}$$

for $i \in [n]$. Finally, it sets $\pi_{n+1} \coloneqq \pi_n^{w_{n+1}}$ and outputs $(Y, \pi = (\pi_1, \ldots, \pi_{n+1}))$.

**Verification:** Given $1^\lambda$, $\mathsf{vk}$, $X \in \{0,1\}^*$ and $(Y, \pi = (\pi_1, \ldots, \pi_{n+1}))$, $\mathsf{Vfy}_{\mathsf{cAHF}}$ tests if $Y$ and $\pi$ contain only valid group elements using $\mathsf{GrpVfy}$ and $\mathsf{GrpElemVfy}$, and outputs 0 if not. Then it defines $\pi_0 \coloneqq g_0$, and outputs 1 if and only if for all $i \in [n]$ it holds that

$$e(\pi_i, g) = \begin{cases} e(\pi_{i-1}, g) & \text{if } H(X)_i = 0 \text{ and} \\ e(\pi_{i-1}, g_i) & \text{if } H(X)_i = 1, \end{cases} \tag{3.22}$$

and both

$$e(\pi_{n+1}, g) = e(\pi_n, g_{n+1}) \qquad \text{and} \qquad Y = e(\pi_{n+1}, h) \tag{3.23}$$

hold.

Comparison to Jager's VRF. Our construction improves Jager's VRF in two regards, one already present in the publication at SAC 2019 [JN19a] and one introduced in [JN19b], the full version of [JN19a].

- First, our verification and secret key contain *only one element* for each $i \in [n]$, compared to two in Jager's VRF. This improvement is possible by encoding the keys of the cAHF into the public and secret keys in a more efficient way. We explain this technique in detail in the proof of Theorem 5. This improvement is not contained in the publication at SAC 2019 [JN19a].

- The second improvement is that we instantiate the VRF with a *cAHF instead of a bAHF.* Therefore, the algorithms $\mathsf{Eval}_{\mathsf{cAHF}}$ and $\mathsf{Vfy}_{\mathsf{cAHF}}$ apply *a standard hash function instead of an error correcting code* to each input. This changes the input space of the VRF from $\{0,1\}^\lambda$ to $\{0,1\}^*$. In the same way, cAHFs are applicable to the VRFs of Katsumata [Kat17] and Yamada [Yam17a]. This improvement is discussed in the SAC 2019 publication [JN19a].

Observe, that the first improvement is also applicable when the VRF is instantiated with a bAHF instead of a cAHF.

*Remark* 7. The terms $w_{n+1}$ and $g_{n+1}$ in our construction are equivalent to appending a bit that is always set to one to each output of the hash function. This ensures that there is no input to the hash function resulting in an all-zero output, which is needed in the security proof. Alternatively, we could assume preimage resistance of the hash function and use that there is no efficient adversary that is able to find a preimage for the all-zeros output. We could then guess a position that is one for all inputs provided by the adversary. We chose to introduce $w_{n+1}$ and $g_{n+1}$ instead for simplicity and clarity.

**Correctness and Unique Provability of the VRF from cAHFs.**

We begin by proving that $\mathcal{VRF}_{\mathsf{cAHF}}$ is indeed correct and provides unique provability. The argumentation mostly follows the respective arguments from [Jag15].

Correctness. For any $\lambda \in \mathbb{N}$ and any $X \in \{0,1\}^*$, let $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}_{\mathsf{cAHF}}(1^\lambda)$ and $(Y, \pi) \leftarrow \mathsf{Eval}_{\mathsf{cAHF}}(\mathsf{sk}, X)$. We now consider the behavior of the verification algorithm $\mathsf{Vfy}_{\mathsf{cAHF}}(1^\lambda, \mathsf{vk}, X, (Y, \pi))$. Since $Y$ and $\pi$ are results of $\mathsf{Eval}_{\mathsf{cAHF}}$, both $\mathsf{GrpVfy}$ and $\mathsf{GrpElemVfy}$ output 1 and thus $\mathsf{Vfy}_{\mathsf{cAHF}}$ does not reject the input. Now let $(H_1, \ldots, H_n) \coloneqq H(X)$ and $\pi_0 \coloneqq g_0$. Then for all $i \in [n]$, we have

$$e(\pi_i, g) = e\left(\pi_{i-1}^{w_i^{H_i}}\right) = \begin{cases} e(\pi_{i-1}^{w_i^0}, g) = e(\pi_{i-1}, g) & \text{if } H_i = 0 \text{ and} \\ e(\pi_{i-1}^{w_i^1}, g) = e(\pi_{i-1}, g^{w_i}) = e(\pi_{i-1}, g_i) & \text{if } H_i = 1. \end{cases}$$

Therefore, Equation (3.22) is fulfilled. Furthermore, we have

$$e(\pi_{n+1}, g) = e(\pi_n^{w_{n+1}}, g) = e(\pi_n, g_{n+1}),$$

which fulfills the first part of Equation (3.23). Moreover, notice that for all $i \in [n]$, we have that

$$\pi_i = g^{w_0 \prod_{j=1}^{i} w_j^{H_j}} \qquad \text{and} \qquad \pi_{n+1} = \pi_n^{w_{n+1}} = g^{w_0 \left(\prod_{j=1}^{i-1} w_j^{H_j}\right) w_{n+1}} = g^{w_X}$$

holds. Thus, $e(\pi_{n+1}, h) = e(g^{w_X}, h) = e(g, h)^{w_X}$ holds, fulfilling Equation (3.23)'s second part. Hence, $\mathsf{Vfy}_{\mathsf{cAHF}}$ outputs 1 on input $(1^\lambda, \mathsf{vk}, X, (Y, \pi))$. Finally, as can easily be verified, $\mathsf{SetupVRF}_{\mathsf{cAHF}}, \mathsf{Eval}_{\mathsf{cAHF}}$ and $\mathsf{Vfy}_{\mathsf{cAHF}}$ are all polynomial-time algorithms.

Unique Provability. We show that for any $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}_{\mathsf{cAHF}}(1^k)$ and $X \in \{0,1\}^*$, there is a unique $Y \in \{0,1\}^*$ such that there exists a proof $\pi$ with $\mathsf{Vfy}_{\mathsf{cAHF}}(1^\lambda, \mathsf{vk}, X, Y, \pi) = 1$. Let $(H_1, \ldots, H_n) \coloneqq H(X)$, then, since elements from $\mathbb{G}$ and $\mathbb{G}_T$ have an unique encoding, $\pi_i = \pi_{i-1}^{(w_i)^{H_i}}$ is the unique group element fulfilling Equation (3.22). By induction, $\pi_i$ is therefore uniquely defined for all $n \in [n]$. Analogously, $\pi_{n+1}$ is uniquely defined by the first part of Equation (3.23). Finally, since $\pi_{n+1}$ is uniquely defined, so is $Y$ by the second part of Equation (3.23).

**Pseudorandomness of $\mathcal{VRF}_{\mathsf{cAHF}}$.**

We will prove the pseudorandomness of our construction based on the $q$-decisional Diffie-Hellman assumption, also used in the proof of pseudorandomness by Jager for his VRF [Jag15], with small $q = \lceil \log(4t_{\mathcal{A}}(2t_{\mathcal{A}} - 1)/\varepsilon_{\mathcal{A}})) \rceil = \mathcal{O}(\log \lambda)$.

**Theorem 5.** *If $\mathcal{VRF}_{\mathsf{cAHF}}$ is instantiated with the family of computational admissible hash functions from Theorem 4, then for any legitimate attacker $\mathcal{A}$ that breaks the* pseudorandomness *of $\mathcal{VRF}_{\mathsf{cAHF}}$ in time $t_{\mathcal{A}}$ with advantage $\varepsilon_{\mathcal{A}} \coloneqq \mathsf{Adv}^{\mathsf{Psrd}}_{\mathcal{VRF}_{\mathsf{cAHF}},\mathcal{A}}(\lambda)$, there exists an algorithm $\mathcal{B}$ that, given (sufficiently close approximations of) $t_{\mathcal{A}}$ and $\varepsilon_{\mathcal{A}}$, breaks the $q$-$\mathsf{DDH}$ assumption with $q = \lceil \log(4t_{\mathcal{A}}(2t_{\mathcal{A}} - 1)/\varepsilon_{\mathcal{A}}) \rceil$ in time $t_{\mathcal{B}} \approx t_{\mathcal{A}}$ and with advantage*

$$\mathsf{Adv}^{q\text{-}\mathsf{DDH}}_{\mathcal{B}}(\lambda) \geq \frac{\varepsilon^2_{\mathcal{A}}}{32t^2_{\mathcal{A}} - 16t_{\mathcal{A}}}.$$

*In particular, if $\mathcal{VRF}_{\mathsf{cAHF}}$ is instantiated with a certified bilinear group generator for which the $q$-$\mathsf{DDH}$ assumption is hard, then $\mathcal{VRF}_{\mathsf{cAHF}}$ is a secure verifiable random function.*

Remarkably, the proof of Theorem 5 is significantly simpler than the corresponding AHF-based proof from [Jag15].

*Proof.* We prove the theorem with a sequence of games as discussed in Section 2.5. In the sequel, we denote the event that Game $i$ outputs 1 by $\mathsf{G}_i$ and denote the adversary's advantage in Game $i$ by $\mathsf{E}_i \coloneqq |\Pr[\mathsf{G}_i] - 1/2|$.

*Game 0.* This is the original VRF security game, as described in Definition 6. By definition, it holds that

$$\mathsf{E}_0 = \mathsf{Adv}^{\mathsf{Psrd}}_{\mathcal{A},\mathcal{VRF}_{\mathsf{cAHF}}}(\lambda).$$

*Game 1.* Recall that Theorem 5 assumes knowledge of (sufficiently close approximations of) the running time $t_{\mathcal{A}}$ and the advantage $\varepsilon_{\mathcal{A}}$. In this game, the challenger additionally runs $\mathsf{K} \xleftarrow{\$} \mathsf{cAdmSmp}(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ in the end of the security experiment but before returning the result of the game. It uses its knowledge of (the sufficiently close approximations of) $t_{\mathcal{A}} \in \mathbb{N}$ and $\varepsilon_{\mathcal{A}} \in [0, 1]$. Note that $\mathsf{K}$ and $H$ now define the function $\mathsf{F}_{\mathsf{K},H}(X)$ from Equation (3.20) and events $\mathsf{coll}, \mathsf{badChal}$ and $\mathsf{badEval}$ as

$$\begin{aligned}
\mathsf{badChal} &\iff \mathsf{F}_{\mathsf{K},H}(X^*) = 0 \\
\mathsf{coll} &\iff \exists\, i, j \text{ with } X^{(i)} \neq X^{(j)} \text{ s.t.} \\
&\qquad \forall\, \ell \in [n] : H(X^{(i)})_\ell = H(X^{(j)})_\ell \vee \mathsf{K}_\ell = \bot \\
\mathsf{badEval} &\iff \exists\, i \in [Q] \text{ s.t. } \mathsf{F}_{\mathsf{K},H}(X^{(i)}) = 1
\end{aligned}$$

like in Definition 19. Note that we denote with $X^{(1)}, \ldots, X^{(Q)} \in \{0,1\}^*$ the evaluation queries made by $\mathcal{A}$ and by $X^*$ the challenge chosen by $\mathcal{A}$. The changes we

made in this game are purely conceptual and perfectly hidden from $\mathcal{A}$. It thus holds that

$$\mathsf{E}_1 = \mathsf{E}_0.$$

*Game 2.* In this game, the challenger checks whether the event $\mathsf{coll}$ occurred after generating $\mathsf{K}$ in the end and outputs a random bit if it occurred. We have that

$$\begin{aligned}
\Pr[\mathsf{coll}] &\geq |\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| && (3.24)\\
&= \left|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2] + \frac{1}{2} - \frac{1}{2}\right| \\
&= \left|\left(\Pr[\mathsf{G}_1] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| \\
&= \left|\left(\Pr[\mathsf{G}_1] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| + \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| - \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| \\
&\geq \left|\left(\Pr[\mathsf{G}_1] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right) + \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| - \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| && (3.25)\\
&= \left|\Pr[\mathsf{G}_1] - \frac{1}{2}\right| - \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| = \mathsf{E}_1 - \mathsf{E}_2,
\end{aligned}$$

where Equation (3.24) follows from Lemma 1 (Shoup's Difference Lemma) and Equation (3.25) follows from the triangle inequality. Rearranging the terms above, we thus obtain that

$$\mathsf{E}_2 \geq \mathsf{E}_1 - \Pr[\mathsf{coll}] = \varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}].$$

*Game 3.* This game proceeds identically to the previous game, except that the challenger checks whether the event $\mathsf{badChal}$ occurred in the end and outputs a random bit if it occurred. We have that

$$\begin{aligned}
\Pr[\mathsf{G}_3] &= \Pr[\mathsf{G}_3 \wedge \mathsf{badChal}] + \Pr[\mathsf{G}_3 \wedge \neg\mathsf{badChal}] \\
&= \Pr[\mathsf{G}_3 \mid \mathsf{badChal}]\,(1 - \Pr[\neg\mathsf{badChal}]) \\
&\quad + \Pr[\mathsf{G}_3 \mid \neg\mathsf{badChal}]\Pr[\neg\mathsf{badChal}] \\
&= 1/2 + \Pr[\neg\mathsf{badChal}]\,(\Pr[\mathsf{G}_3 \mid \neg\mathsf{badChal}] - 1/2) && (3.26)\\
&= 1/2 + \Pr[\neg\mathsf{badChal}]\,(\Pr[\mathsf{G}_2 \mid \neg\mathsf{badChal}] - 1/2) && (3.27)\\
&= 1/2 + \Pr[\neg\mathsf{badChal}]\,(\Pr[\mathsf{G}_2] - 1/2), && (3.28)
\end{aligned}$$

where the Equation (3.26) follows from $\Pr[\mathsf{G}_3 \mid \mathsf{badChal}] = 1/2$, since a random bit is returned if $\mathsf{badChal}$ occurs, Equation (3.27) uses that by definition of the games it holds that $\Pr[\mathsf{G}_3 \mid \neg\mathsf{badChal}] = \Pr[\mathsf{G}_2 \mid \neg\mathsf{badChal}]$, and the Equation (3.28) uses

$\Pr[\mathsf{G}_2 \mid \neg\mathsf{badChal}] = \Pr[\mathsf{G}_2]$, since Game 2 is independent of $\mathsf{badChal}$. This is because $\mathsf{K}$ is only sampled after $\mathcal{A}$ made all its queries and stated its challenge and $\mathsf{K}$ is therefore perfectly hidden from $\mathcal{A}$. We thus conclude that

$$\begin{aligned}\mathsf{E}_3 = \left|\mathsf{G}_3 - \frac{1}{2}\right| &= \left|\Pr[\neg\mathsf{badChal}] \cdot \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| = \Pr[\neg\mathsf{badChal}] \cdot \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| \\ &= \Pr[\neg\mathsf{badChal}] \cdot \mathsf{E}_2.\end{aligned}$$

*Game 4.* In this game the challenger also checks for the event $\mathsf{badEval}$ in the end and aborts if it occurs and outputs a random bit. Since we have that $\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal}$ holds by the properties of cAHFs (see Definition 19), this is a purely conceptual change and it holds that

$$\mathsf{E}_4 = \mathsf{E}_3.$$

*Game 5.* We replace the events $\mathsf{badChal}, \mathsf{coll}$ and $\mathsf{badEval}$ with an equivalent event $\mathsf{bad}$, in order to simplify the construction of adversary $\mathcal{B}$. We let $\mathsf{bad}$ denote the event that $\mathsf{coll} \vee \mathsf{badChal} \vee \mathsf{badEval}$. Since this is a purely conceptual change, it holds that

$$\mathsf{E}_5 = \mathsf{E}_4.$$

*Game 6.* In this game, we change the challenger to generate $\mathsf{K}$ in the very beginning instead of in the end. Furthermore, the challenger already sets the event $\mathsf{bad}$ as soon as the adversary makes an evaluation query or states challenge that causes the event. However, the challenger still only aborts and outputs a random bit if $\mathsf{bad}$ occurred in the very end. Therefore, these are purely conceptual changes and it holds that

$$\mathsf{E}_6 = \mathsf{E}_5.$$

*Game 7.* In this game, the challenger aborts and outputs a random bit as soon as it occurs. Because Game 6 and Game 7 are *identical until bad*, it holds that

$$\mathsf{E}_7 = \mathsf{E}_6.$$

*Game 8.* In this game, the challenger always returns $Y_1 \xleftarrow{\$} \mathbb{G}_T$, regardless of the value of $b$ in $G^{\mathsf{Psrd}}$. We claim that there is an algorithm $\mathcal{B}$ such that

$$|\mathsf{E}_7 - \mathsf{E}_8| = \mathsf{Adv}^{q\text{-}\mathsf{DDH}}_{\mathcal{B}}(\lambda) \tag{3.29}$$

for $q \coloneqq \lceil \log(4t_{\mathcal{A}}(2t_{\mathcal{A}} - 1)/\varepsilon_{\mathcal{A}}) \rceil$. We proceed by describing $\mathcal{B}$.

Initialization. The input of $\mathcal{B}$ is the $q$-DDH-challenge $(\mathcal{BG}, \tilde{g}, h, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^q}, h, T)$. $\mathcal{B}$ first checks whether $\tilde{g}^{\alpha} \stackrel{?}{=} 1_{\mathbb{G}}$. Since $\tilde{g}$ is a generator of $\mathbb{G}$, this holds if and only if $\alpha = 0$. In this case, $\mathcal{B}$ also checks whether $T \stackrel{?}{=} 1_{\mathbb{G}_T}$ and outputs 0 if the statement is true and 1 otherwise. This is the correct solution to the $q$-DDH-challenge if $\alpha = 0$, because if $\alpha = 0$, then $e(g,h)^{\alpha^{q+1}} = e(g,h)^0 = 1_{\mathbb{G}_T}$. Therefore, we assume $\alpha \neq 0$ for the remainder of the proof. Moreover, $\mathcal{B}$ draws a random bit $b \xleftarrow{\$} \{0,1\}$.

Generating the verification key. After receiving the values from the $q$-DDH-challenge, $\mathcal{B}$ first samples $H \xleftarrow{\$} \mathcal{H}$ and $\mathsf{K} \xleftarrow{\$} \mathsf{cAdmSmp}(1^{\lambda}, t_{\mathcal{A}}, \varepsilon_A)$. It then computes the values $g_i$ exactly as in the original $\mathsf{SetupVRF}_{\mathsf{cAHF}}$-algorithm for most $i \in [n]$ by choosing $w_i \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}$ and setting $g_i \coloneqq g^{w_i}$, but with the exception that

$$g_i \coloneqq \begin{cases} g^{w_i + 1/\alpha} & \text{if } \mathsf{K}_i = 0 \\ g^{w_i + \alpha} & \text{if } \mathsf{K}_i = 1 \end{cases}$$

for all $i \in [n]$ with $\mathsf{K}_i \neq \perp$. Recall that due to our choices in Theorem 5, we have that $\sigma(\mathsf{K}) \leq q$. Therefore, $\mathcal{B}$ can compute $g^{w_0 + \alpha^{q - \sigma(\mathsf{K}) - 1}}$ since $q - \sigma(\mathsf{K}) - 1$ is at least $-1$ and at most $q - 1$, and $g^{1/\alpha} = \tilde{g}, \ldots, g^{\alpha^{q-1}} = \tilde{g}^{\alpha^q}$ are part of the $q$-DDH-challenge. $\mathcal{B}$ can compute $g^{w_i + 1/\alpha}$ for the same reason. Moreover, note that all values $g_i$ are distributed exactly as in the original security experiment. This way of generating $\mathsf{vk}$ is *the main difference* from this construction compared to the construction published at SAC 2019 [JN19a] and Jager's VRF [Jag15]. In both earlier constructions, there were $w_{i,0}$ and $w_{i,1}$ for each $i \in [n]$ instead of only $w_i$ in this construction. The former constructions then added $\alpha$ to $w_{i,\mathsf{K}_i}$ if $\mathsf{K}_i \neq \perp$. Adding either $\alpha$ or $1/\alpha$ depending on $\mathsf{K}_i$ allows us to include only a single element for each $i \in [n]$ instead of two.

Helping definitions. To describe how $\mathcal{B}$ responds to evaluation queries and $\mathcal{A}$'s challenge, we define four sets $I_{v,X}$, $I^0_{v,X}$, $I^1_{v,X}$ and $I^{\perp}_{v,X}$, which depend on a VRF input $X \in \{0,1\}^*$ and an integer $v \in [n]$. $I_{v,X} \subseteq [v] \subseteq [n]$ is the set of all indices such that $H(X)_i = 1$. The other sets partition $I_{v,X}$ in the respective subsets of

indices where $\mathsf{K}_i$ is $0, 1$ or $\perp$. Formally, the sets are defined as follows.

$$\begin{aligned}
I_{v,X} &\coloneqq \{i \in [v] : H(X)_i = 1\},\\
I^0_{v,X} &\coloneqq \{i \in [v] : H(X)_i = 1 \wedge \mathsf{K}_i = 0\},\\
I^1_{v,X} &\coloneqq \{i \in [v] : H(X)_i = 1 \wedge \mathsf{K}_i = 1\} \text{ and}\\
I^\perp_{v,X} &\coloneqq \{i \in [v] : H(X)_i = 1 \wedge \mathsf{K}_i = \perp\}
\end{aligned}$$

We observe that $I_{v,X} = I^0_{v,X} \cup I^1_{v,X} \cup I^\perp_{v,X}$. Based on these sets, we define polynomials $\mathsf{P}_{v,X}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ for all $X \in \{0,1\}^*$ and $v \in [n+1]_0$. We let $\mathsf{P}_{0,X}(\mathsf{Z}) \coloneqq w_0 + \mathsf{Z}^{q-\sigma(\mathsf{K})-1}$ and define

$$\mathsf{P}_{v,X}(\mathsf{Z}) = \begin{cases} \mathsf{P}_{v-1,X}(\mathsf{Z}) \cdot (w_i + 1/\mathsf{Z}) & \text{if } v \in I^0_{v,X},\\ \mathsf{P}_{v-1,X}(\mathsf{Z}) \cdot (w_i + \mathsf{Z}) & \text{if } v \in I^1_{v,X},\\ \mathsf{P}_{v-1,X}(\mathsf{Z}) \cdot w_i & \text{if } v \in I^\perp_{v,X} \text{ and}\\ \mathsf{P}_{v-1,X}(\mathsf{Z}) & \text{otherwise.} \end{cases}$$

for all $X \in \{0,1\}^*$ and $v \in [n]$. Finally, we define $\mathsf{P}_{n+1,X}(\mathsf{Z}) \coloneqq \mathsf{P}_{n,X}(\mathsf{Z}) \cdot (w_{n+1} + \mathsf{Z})$ for all $X \in \{0,1\}^*$. We observe some important properties of these helping definitions in the following lemma.

**Lemma 5.** *Let $I^0_{v,X}$ and $I^1_{v,X}$ be as above, then*

$$\left|I^0_{v,X}\right| \leq q - \sigma(\mathsf{K}) \qquad \text{and} \qquad \left|I^1_{v,X}\right| \leq \sigma(\mathsf{K})$$

*holds for all $v \in [n]$. Furthermore, for $\mathsf{P}_{v,X}$ as above, it holds that*

$$\mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) = q - \sigma(\mathsf{K}) - 1 - \left|I^0_{v,X}\right| + \left|I^1_{v,X}\right|$$

*for all $v \in [n]$, where $\mathsf{deg}$ denotes the degree of a polynomial. In particular, we have*

$$-1 \leq \mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) \leq q - 1$$

*for all $v \in [n]$. Note that we simplify notation by writing $\mathsf{deg}(f(\mathsf{Z})) = -k$ for a rational function $f$ such that $1/f(\mathsf{Z})$ is a polynomial of degree $k \in \mathbb{N}$.*

*Proof of Lemma 5.* Observe that by the Definition of $\mathsf{cAdmSmp}$ in Theorem 4 and the choice of $q$ in Theorem 5 we have $\sigma(\mathsf{K}) \leq q$. Furthermore, $I^1_{n,X}$ contains up to one element for each $i \in [n]$ such that $\mathsf{K}_i = 1$, which are at most $\sigma(\mathsf{K})$ many. Analogously, $I^0_{v,X}$ can contain at most $q - \sigma(\mathsf{K})$ elements since $\mathsf{K}$ has only $q - \sigma(\mathsf{K})$ positions where it is 0. We proceed by proving that

$$\mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) = q - \sigma(\mathsf{K}) - 1 - \left|I^0_{v,X}\right| + \left|I^1_{v,X}\right| \tag{3.30}$$

holds for all $v \in [n]$. Observe that in the definition of $\mathsf{P}_{v,X}$ each $i \in I^0_{v,X}$ adds a factor $(w_i + 1/\mathsf{Z})$ to $\mathsf{P}_{v-1,X}(\mathsf{Z})$ and by that decreases the degree of the polynomial by one. Analogously, each element in $I^1_{v,X}$ increases the degree of the polynomial

by one. Finally, $\mathsf{P}_{0,X}(\mathsf{Z})$ is defined as $(w_0 + \mathsf{Z}^{q-\sigma(\mathsf{K})-1})$ yielding, together with the upper bounds on $|I_{v,X}|^0$ and $|I_{v,X}|^1$ above, Equation (3.30).
We finish the proof of Lemma 5, by showing that

$$-1 \leq \mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) \leq q-1$$

holds for all $v \in [n]$. By Equation (3.30) and $|I_{v,X}| \leq \sigma(\mathsf{K})$, the following holds.

$$\begin{aligned}\mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) &= q - \sigma(\mathsf{K}) - 1 - \left|I^0_{v,X}\right| + \left|I^1_{v,X}\right| \\ &\leq q - \sigma(\mathsf{K}) - 1 + \left|I^1_{v,X}\right| \\ &\leq q - \sigma(\mathsf{K}) - 1 + \sigma(\mathsf{K}) = q - 1.\end{aligned}$$

We conclude the lower bound analogously by applying Equation (3.30) and $|I_{v,X}| \leq q - \sigma(\mathsf{K})$.

$$\begin{aligned}\mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) &= q - \sigma(\mathsf{K}) - 1 - \left|I^0_{v,X}\right| + \left|I^1_{v,X}\right| \\ &\geq q - \sigma(\mathsf{K}) - 1 - \left|I^0_{v,X}\right| \\ &\geq q - \sigma(\mathsf{K}) - 1 - q - \sigma(\mathsf{K}) = -1\end{aligned}$$

□

Now, having Lemma 5 at hand, we proceed with the proof of Theorem 5 and, similar to [Jag15], make the following observations.

1. For all $X$ with $\mathsf{F}_{\mathsf{K},H}(X) = 0$ and $v \in [n+1]_0$, we have $-1 \leq \mathsf{deg}(\mathsf{P}_{v,X}(\mathsf{Z})) \leq q-1$. Note that in contrast to Lemma 5, the bound also holds for $v = n+1$. For this, observe that for all $X$ with $\mathsf{F}_{\mathsf{K},H}(X) = 0$, there is an $i \in [n]$ such that $\mathsf{K}_i \neq \perp$ and $\mathsf{K}_i \neq H(X)_i$. Therefore, at least one of the following conditions hold.

   $$\left|I^0_{v,X}\right| > 0 \text{ for all } v \geq i \qquad \left|I^1_{v,X}\right| < q - \sigma(\mathsf{K}) \text{ for all } v \in [n]$$

   By Lemma 5, we thus have $\mathsf{deg}(\mathsf{P}_{n,X}(\mathsf{Z})) \leq q-2$, which implies that the degree of $\mathsf{P}_{n+1,X}(\mathsf{Z})$ is at most $q-1$. Hence, $\mathcal{B}$ can efficiently compute $g^{\mathsf{P}_{v,X}(\alpha)} = \tilde{g}^{\alpha \cdot \mathsf{P}_{v,X}(\alpha)}$ for all $v \in [n+1]_0$. To this end, $\mathcal{B}$ would first computes the coefficients $\gamma_0, \ldots, \gamma_q$ of the polynomial $\mathsf{Z} \cdot \mathsf{P}_{v,X}(\mathsf{Z}) = \sum_{i=0}^{q} \gamma_i \mathsf{Z}^i$ with degree at most $q$, and then compute

   $$\tilde{g}^{\alpha \cdot \mathsf{P}_{v,X}(\alpha)} := \tilde{g}^{\sum_{i=0}^{q} \gamma_i \alpha^i} = \prod_{i=0}^{q} (\tilde{g}^{\alpha^i})^{\gamma_i}$$

   using the terms $(\tilde{g}, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^q})$ from the $q$-$\mathsf{DDH}$ challenge.

2. If $\mathsf{F}_{\mathsf{K},H}(X) = 1$, then $H(X)_i = \mathsf{K}_i$ for all $i \in [n]$ with $\mathsf{K}_i \neq \perp$ and it thus holds that $\left|I^0_{n,X}\right| = 0$ and $\left|I^1_{n,X}\right| = \sigma(\mathsf{K})$. By Lemma 5, $\mathsf{Z} \cdot \mathsf{P}_{n+1,X}(\mathsf{Z})$ therefore has degree $q+1$. We do not know how $\mathcal{B}$ can efficiently compute $g^{\mathsf{P}_{n+1,X}(\alpha)} = \tilde{g}^{\alpha \cdot \mathsf{P}_{n+1,X}(\alpha)}$ in this case.

Responding to evaluation queries. For an evaluation query $X \in \{0,1\}^*$, $\mathcal{B}$ first computes $\mathsf{F}_{\mathsf{K},H}(X)$. Just as the challenger in Game 8 and Game 7, $\mathcal{B}$ aborts and outputs a random bit if $\mathsf{F}_{\mathsf{K},H}(X) = 1$. Moreover, $\mathcal{B}$ checks for each query whether the event $\mathsf{coll}$ occurred and if so outputs a random bit just as the challenger in Game 8 and Game 7. If $\mathsf{F}_{\mathsf{K},H}(X) = 0$, then $\mathcal{B}$ sets

$$\pi_i \coloneqq g^{\mathsf{P}_{i,X}(\alpha)}$$

for all $i \in [n+1]_0$ and

$$Y \coloneqq e(\pi_{n+1}, h),$$

which it can compute as we outlined above. Observe that $\mathcal{B}$'s response to $\mathcal{A}$'s evaluation query is distributed exactly as in Game 8 and Game 7.

Responding to $\mathcal{A}$'s challenge. Just as the challenge in Game 8 and Game 7, checks whether the event $\mathsf{bad}$ occurred and aborts and outputs a random bit if it is the case. We observe that if $\mathsf{bad}$ did not occur, then we have that $\mathsf{F}_{\mathsf{K},H}(X^*) = 1$ holds. Now, $\mathcal{B}$ sets

$$Y^* \xleftarrow{\$} \mathbb{G}_T \text{ if } b = 1$$

and it computes

$$Y^* \coloneqq T^{\gamma_{q+1}} \cdot \prod_{i=0}^{q} e((\tilde{g}^{\alpha^i})^{\gamma_i}, h) = T^{\gamma_{q+1}} \cdot e(\tilde{g}^{\sum_{i=0}^{q} \gamma_i \alpha^i}, h) \text{ if } b = 0$$

where $\gamma_0, \ldots, \gamma_{q+1}$ are the coefficients of the degree-$(q+1)$-polynomial $\mathsf{Z}\cdot\mathsf{P}_{n+1,X^*}(\mathsf{Z}) = \sum_{i=0}^{q+1} \gamma_i x^i$. Note that if $T$ is uniformly random in $\mathbb{G}_T$, then $Y^*$ is always uniformly random, regardless of the value of $b$, and is thus distributed exactly as in Game 8. However, if $T = e(\tilde{g}, h)^{\alpha^{q+1}}$, then $Y^*$ is uniformly random if $b = 1$ and is $e(g,h)^{w_{X^*}}$ if $b = 0$. In particular, if $T = e(\tilde{g}, h)^{\alpha^{q+1}}$, then $Y^*$ is distributed exactly as in Game 7.

Solving the $q$-$\mathsf{DDH}$ challenge. Finally, when $\mathcal{A}$ states its guess $b' \in \{0,1\}$, then $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise.

$\mathcal{B}$'s running time. The running time $t_{\mathcal{B}}$ of $\mathcal{B}$ consists of the running time $t_{\mathcal{A}}$ of $\mathcal{A}$ plus the time required to answer $\mathcal{A}$'s queries. The latter step essentially consists of the operations defined in the construction of the VRF in Section 3.3.2 plus minor operations like sampling $\mathsf{K}$, evaluating $\mathsf{F}_{\mathsf{K},H}$, checking for collisions, calculate the $\gamma_i$ to compute $g^{\mathsf{P}_{v,X}(\alpha)}$ and some group operations to compute the $g_i$. Since we include the runtime of the experiment in the adversary's runtime, we have that have $t_{\mathcal{B}} \approx t_{\mathcal{A}}$.

$\mathcal{B}$'s advantage. As we noted throughout the description of $\mathcal{B}$, all responses $\mathcal{B}$ gives to $\mathcal{A}$ are distributed exactly as in Game 8 if $T \xleftarrow{\$} \mathbb{G}_T$ and are distributed exactly as in Game 7 if $T = e(\tilde{g}, h)^{\alpha^{q+1}}$. We thus conclude that

$$\mathsf{Adv}_{\mathcal{B}}^{q\text{-}\mathsf{DDH}}(\lambda) = |\mathsf{E}_7 - \mathsf{E}_8|$$

holds as we claimed in Equation (3.29). Summing up probabilities from Game 0 to Game 8, we thus obtain

$$\begin{aligned} \Pr[E_8] &\geq 1/2 + \Pr[\neg\mathsf{badChal}] \left(\Pr[\mathsf{E}_0] - 1/2 - \Pr[\mathsf{coll}]\right) \\ &= 1/2 + \Pr[\neg\mathsf{badChal}] \left(\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]\right) \\ &\geq 1/2 + \tau(\lambda) \end{aligned} \tag{3.31}$$

for some non-negligible function $\tau(\lambda)$, where the last inequality is due to the definition of cAHFs (see Equation (3.21)). In particular, when instantiated concretely with the computational AHF from Theorem 4, then we have that

$$\mathsf{Adv}_{\mathcal{B}}^{q\text{-}\mathsf{DDH}}(\lambda) \geq \varepsilon_{\mathcal{A}}^2 / (32 t_{\mathcal{A}}^2 - 16 t_{\mathcal{A}})$$

holds as claimed. □

## 3.4 More Efficient Verifiable Random Functions from Blockwise Partitioning

We saw in the previous section how we can construct VRFs from cAHFs instead of bAHFs, to replace a component with information-theoretic security with a more efficient component with computational security. In our definition of cAHFs in Definition 19, we took care to define cAHFs such that they can be used as a drop-in replacement for bAHFs. In this section, we abandon this design goal and instead aim to construct VRFs that are as efficient as possible. Throughout this section, we will first describe *blockwise partitioning* as a semi-generic technique to achieve adaptive security in Section 3.4.1 and then show how it can be used to construct more efficient VRFs in Section 3.4.2. Note that we will show in Chapter 5 how blockwise partitioning can also be used to construct IB-KEMs (see Definition 7) from lattices.

### 3.4.1 Blockwise Partitioning via Near-Collision Resistance

We approach this goal by improving our technique from the previous section by combining it with *confined guessing* [BHJ+13, BHJ+15], a semi-generic technique to construct efficient and adaptively-secure digital signature schemes. Apart from its introduction in [BHJ+13, BHJ+15], confined guessing has been used for instance in [DM14, Alp15] and with a slight variation in [JK18]. Unfortunately, confined

guessing achieves only non-adaptive security, which is sufficient for signatures (as adaptive security can then be easily achieved generically by using chameleon hash functions [KR00]). However, it is therefore neither applicable to IB-KEMs, nor to VRFs[3].

We propose *blockwise partitioning* as a new semi-generic technique, and show how it can be used to construct efficient VRFs with adaptive security. It is based on the *near-collision resistance* of a cryptographic hash function and is similar in spirit to the closely related notion of truncation collision resistance [JK18], which we used in the previous section.

High-level perspective. In order to sketch the main idea, consider VRFs as an example. The general approach of confined guessing is analogous to our construction of cAHFs from TCR. That is, we let the reduction guess $\eta = \lceil \log(4t(2t-1)/\varepsilon) \rceil$ many bits of $H(X^*)$, where $H$ is a cryptographic hash function and $X^*$ is the challenge input chosen by the adversary. Using standard techniques from selectively-secure constructions, we prove security with a reduction which is successful if the $\eta$ bits of $H(X^*)$ are guessed correctly, while the hash of every input for which the adversary requests the evaluation of the VRF $X^{(i)}$ differs in at least one of the $\eta$ bits.

In contrast to the constructions in the previous section, constructions employing the confined guessing approach, such as used in [JK18], only use powers of two[4] as $\eta$. That is, they would use the smallest power of two that is larger or equal to $\lceil \log(4t(2t-1)/\varepsilon) \rceil$ as $\eta$ instead of $\lceil \log(4t(2t-1)/\varepsilon) \rceil$ directly. Ultimately, this leaves only $\mathcal{O}(\log(\lambda))$ many possibilities for $\eta$ and consequently enables constructions with smaller keys and signatures. However, this also leads to $\eta$ being almost twice of what would be ideal and incurs an additional quadratic security loss and also requires a stronger $q$-type assumption with quadratically larger $q$.

We address this issue by viewing $\eta = \lceil \log(4t(2t-1)/\varepsilon) \rceil$ as a sum of powers of two and the output of the hash function as the concatenation of blocks of exponentially growing length, *i.e.* the first bit is the first block, bits two and three are the second block, bits four to seven are the third block, and so on. Our reduction then uses the ideal choice for $\eta$ and guesses the bits in the blocks whose lengths sum up to exactly $\eta$. Compared to confined guessing, this more fine-grained guessing yields constructions with tighter security from weaker assumptions. It also reduces the required output length of the hash function from $4(\lambda+1)$ bits in [JK18] to only $2\lambda+3$ bits. Note that this is essentially optimal for a collision resistant hash function. In particular, for many practical constructions one would use a collision resistant hash function, anyway, to map long inputs to short strings. We compare our

[3]The variation of confined guessing in [JK18] enables the construction of IB-KEMs. We follow a similar approach.

[4]Confined guessing was originally introduced for powers of arbitrary positive numbers and not only two [BHJ+13, BHJ+15]. However, we only describe the approach with basis two for clarity.

techniques to the ones of [JK18] in more detail after formally introducing blockwise partitioning.

We proceed describing the framework and assumptions for blockwise partitioning, give some more technical intuition, and state and prove a technical lemma that will be useful that will be useful in constructing security proofs using blockwise partitioning, in a way that is as modular as possible.

**Blockwise partitioning.**

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function. We will assume in the sequel that $n = \sum_{i=0}^{\ell} 2^i$ for simplicity and ease of exposition. One can generalize this to arbitrary $n$, but this would make the notation rather cumbersome without providing additional insight or clarity. We can then view the output space $\{0,1\}^n$ of the hash function as a direct product of sets of exponentially-increasing size

$$\{0,1\}^n = \{0,1\}^{2^0} \times \cdots \times \{0,1\}^{2^\ell}.$$

For a hash function $H$ we define functions $H_0, \ldots, H_\ell$ such that

$$H_i : \{0,1\}^* \to \{0,1\}^{2^i} \qquad \text{and} \qquad H(x) = H_0(x) \parallel \cdots \parallel H_\ell(x).$$

One can consider each $H_i(x)$ as one "block" of $H(x)$. Note that blocks have exponentially increasing size and there are $\lfloor \log n \rfloor + 1$ blocks in total. In order to avoid confusion, we point out that this notation is close to the notation we used in the previous section, where we used $H(X)_i$ to denote the $i$th bit of $H(X)$. Furthermore, observe that truncation-collision resistance of hash functions is no longer sufficient because it only considers prefixes of $H$ whereas the blocks $H_i$ of $H$ are, except for $H_0$, not prefixes of $H$. We will thus introduce *weak near-collision resistance* of hash functions as a slightly stronger but still quite plausible assumption for hash functions.

Using blockwise partitioning. Let $t = t(\lambda)$ be a polynomial and let $\varepsilon = \varepsilon(\lambda)$ be a non-negligible function such that $\varepsilon > 0$ and $t/\varepsilon < 2^\lambda$ for all $\lambda$. Like in the previous section, think of $t$ and $\varepsilon$ as (approximations of) the running time and advantage of an adversary in a security experiment. We again define an integer $\eta$ depending on $(t, \varepsilon)$ as

$$\eta := \left\lceil \frac{4t \cdot (2t-1)}{\varepsilon} \right\rceil \tag{3.32}$$

Recall that if $n \geq 2\lambda + 3$, then we have $0 \leq \eta \leq n$ as we have shown in Lemma 4. Furthermore, observe that $\eta$ uniquely determines an *index set* $\mathcal{I} = \{i_1, \ldots, i_\omega\} \subseteq \{0, \ldots, \ell\}$, for $\ell = \lfloor \log n \rfloor$, such that $\eta = \sum_{i \in \mathcal{I}} 2^i$. As for cAHFs, the key point in defining $\eta$ as in Equation (3.32) is that it provides the following two properties simultaneously:

The algorithm $\mathsf{BPSmp}(1^\lambda, t, \varepsilon)$

$\eta \coloneqq \lceil 4t(2t-1)/\varepsilon \rceil$
$\ell \coloneqq \lfloor \log(2\lambda + 3) \rfloor$
Let $\mathcal{I} \subseteq [\ell]_0$ s. t. $\eta = \sum_{i=0}^{\ell} 2^i$
**for** $i \in \mathcal{I}$
  $\mathsf{K}_i \xleftarrow{\$} \{0,1\}^{2^i}$
**for** $i \in [\ell]_0 \setminus \mathcal{I}$
  $\mathsf{K}_i \coloneqq \perp$
$\mathsf{K} \coloneqq (\mathsf{K}_0, \ldots, \mathsf{K}_\ell)$
**return** $\mathsf{K}$

*Figure 3.1:* The specification of the algorithm $\mathsf{BPSmp}$ with inputs $1^\lambda$, $t \in \mathbb{N}$ and $\varepsilon$ for $t/\varepsilon < 2^\lambda$.

**Guessing a from polynomially-bounded range.** In order to enable a reduction from adaptive to selective security, we will later have to "predict" a certain hash value $H(X^*)$. Think of $X^*$ as the challenge input in the $G^{\mathsf{Psrd}}$-experiment (see Definition 6) for the security of VRFs. Blockwise partitioning enables this, as we now show. Consider the probabilistic algorithm $\mathsf{BPSmp}$, which is depicted in Figure 3.1. It takes as input $\lambda$, $t$, and $\varepsilon$, computes $\eta$ as in Equation (3.32) and by that defines $\mathcal{I}$ as above, chooses $\mathsf{K}_i \xleftarrow{\$} \{0,1\}^{2^i}$ uniformly at random for $i \in \mathcal{I}$ and defines $\mathsf{K}_i = \perp$ for all $i \notin \mathcal{I}$. Then it outputs

$$(\mathsf{K}_0, \ldots, \mathsf{K}_\ell) \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t, \varepsilon).$$

The joint range of all hash functions $H_i$ with $i \in \mathcal{I}$ is $\{0,1\}^{2^{i_1}} \times \cdots \times \{0,1\}^{2^{i_{|\mathcal{I}|}}}$, which has size

$$2^\eta = 2^{\sum_{i \in \mathcal{I}} 2^i}.$$

Hence, we have that

$$\Pr\left[H_i(X^*) = \mathsf{K}_i \text{ for all } i \in \mathcal{I}\right] = 2^{-\eta}.$$

Note that $2^{-\eta}$ is non-negligible, due to the definition of $\eta$ in Equation (3.32).

**Upper bound on the collision probability.** In Lemma 6 below we will show that near-collision resistance of $H$ guarantees that the probability that an adversary running in time $t$ outputs any two values $x \neq x'$ such that

$$H_i(x) = H_i(x') \qquad \text{for all } i \in \mathcal{I} \tag{3.33}$$

is at most $\varepsilon/2$. Think of $x$ and $x'$ as values chosen adaptively by an adversary in a security experiment. In the context of VRFs these would be chosen

inputs, in context of digital signatures these would be chosen messages, for instance. Note that we do not argue that there is a *negligible* collision probability. This is not possible, because we consider a polynomially-bounded space, where an adversary will always be able to find collisions with non-negligible probability. However, we can guarantee that there will be no collision with probability at least $\varepsilon/2$. This means that an adversary that runs in some time $t$ and has some advantage $\varepsilon$ will sufficiently often be successful *without* finding a collision.

Hence, similar to confined guessing [BHJ+13, BHJ+15] and computational admissible hash functions, blockwise partitioning enables us to guess challenge identities from a sufficiently small space such that the guess is correct with a non-negligible probability. At the same time, it ensures that the space is large enough such that the adversary produces two colliding inputs with probability at most $\varepsilon/2$. Hence, the any adversary breaking a considered cryptosystem with some advantage $\varepsilon$ must "sufficiently often" be successful without finding a collision. Observe that $2^{-\eta}$ can be super-polynomial if $1/\varepsilon$ is super-polynomial, which is not precluded by requiring $\varepsilon$ to be non-negligible.

Blockwise partitioning via near-collision resistance. We will now give a formal definition of weak near-collision resistance and then provide a technical lemma, which will be useful for security proofs based on blockwise partitioning of hash function outputs. Note that weak near-collision resistance is only required for the security of our constructions and we hence only require this property in the respective theorems and not in the constructions themselves. Furthermore, as for cAHFs, we consider a single family of hash functions and thus do not use a security parameter in the definition of weak near-collision resistant hash functions to better capture an instanitation with already standardized hash functions.

**Definition 21** (Weak near-collision resistance)**.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions. For $\eta \in \{1,\dots,n\}$, we say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ breaks the weak $\eta$-near-collision resistance of $\mathcal{H}$, if it runs in time $t_{\mathcal{A}}$ and it holds that

$$\Pr\left[\eta\text{-}\mathsf{wNCR}^{\mathcal{H}}_{\mathcal{A}} = 1\right] \geq t_{\mathcal{A}}(t_{\mathcal{A}} - 1)/2^{\eta+1},$$

where $\eta$-wNCR is the experiment defined in Figure 3.2 and the probability is over the randomness of $\mathcal{A}$ and choosing $H$. We say that $\mathcal{H}$ is *weak near-collision resistant* (wNCR), if there exists no adversary $\mathcal{A}$ breaking the weak $\eta$-near-collision resistance of $\mathcal{H}$ for any $\eta \in \{1,\dots,n\}$.

The following lemma will be useful to apply blockwise partitioning in security proofs.

**Lemma 6.** *Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function, $t$ be a polynomial, and let $\varepsilon$ be a non-negligible function such that $\varepsilon > 0$ and $t/\varepsilon < 2^\lambda$ for all $\lambda$. Let $\eta :=$*

$G^{\eta\text{-wNCR}}_{\mathcal{A},\mathcal{H}}$

$(\mathcal{J}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\eta)$
$H \xleftarrow{\$} \mathcal{H}$
$(X^{(1)}, \ldots, X^{(Q+1)}) \xleftarrow{\$} \mathcal{A}_2(H, \mathsf{st})$
**if** $|\mathcal{J}| = \eta$ **and** $\exists\, x \neq y \in \{X^{(1)}, \ldots, X^{(Q+1)}\}$ **with** $H(x)_i = H(y)_i$ **for all** $i \in \mathcal{J}$ :
  **return** 1
**else**
  **return** 0

*Figure 3.2:* The security experiment for weak near-collision resistance, executed with a family of hash functions $\mathcal{H}$ and adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ outputs an index set $\mathcal{J} \subseteq [n]$ and $\mathcal{H} \subseteq \{h : \{0,1\}^* \to \{0,1\}^n\}$. We restrict $\mathcal{A}_1$ to only output index sets $\mathcal{J}$ with $|\mathcal{J}| = \eta$. Note that $H(x)_i$ denotes the $i$-th bit of $H(x)$.

$\lceil \log(4t \cdot (2t-1)/\varepsilon) \rceil$ *as in Equation* (3.32) *and define set* $\mathcal{I}$ *such that* $\eta = \sum_{i \in \mathcal{I}} 2^i$. *Let* $\mathcal{A}$ *be an algorithm that outputs* $(X^{(1)}, \ldots, X^{(Q)}, X^*)$ *and runs in time* $t$ *and let*

$$(\mathsf{K}_0, \ldots, \mathsf{K}_\ell) \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t, \varepsilon),$$

*where* BPSmp *is the algorithm depicted in Figure 3.1.*

1. *Let* coll *be the event that there exists* $x, x' \in \{X^{(1)}, \ldots, X^{(Q)}, X^*\}$ *such that*

$$H_i(x) = H_i(x') \text{ for all } i \in \mathcal{I}. \tag{3.34}$$

   *Let* badChal *be the event that there exists* $i \in \mathcal{I}$ *such that* $\Pr[H_i(X^*) \neq \mathsf{K}_i]$. *If* $H$ *is drawn uniformly at random from a family of weak near-collision resistant hash functions in the sense of Definition 21, then we have*

$$(\varepsilon - \Pr[\mathsf{coll}]) \cdot \Pr[\neg \mathsf{badChal}] \geq \frac{\varepsilon^2}{32t^2 - 16t}.$$

   *Moreover,* coll *and* badChal *are independent of each other.*

2. *Let* badEval *be the event that there exists* $x \in \{X^{(1)}, \ldots, X^{(Q)}\}$ *with* $x \neq X^*$ *such that* $H_i(x) = \mathsf{K}_i$ *for all* $i \in \mathcal{I}$. *Then we have*

$$\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal}.$$

*Proof.* The proof uses the same inequalities from Lemma 4 that we already used in the context of cAHFs in Section 3.3.

$$\eta \in \{1, \ldots, 2\lambda + 3\}, \qquad \frac{2t(2t-1)}{2^\eta} \leq \frac{\varepsilon}{2} \qquad \text{and} \qquad \frac{1}{2^\eta} \geq \frac{\varepsilon}{16t^2 - 8t} \tag{3.35}$$

We start to prove Property 1 by showing $\Pr[\mathsf{coll}] < \varepsilon/2$. Assume an algorithm $\mathcal{A}$ running in time $t_{\mathcal{A}}$ that outputs $(X^{(1)}, \ldots, X^{(Q)}, X^*)$ such that there exist $x, x' \in \{X^{(1)}, \ldots, X^{(Q)}, X^*\}$ and that Equation (3.34) holds with probability at least $\varepsilon/2$. By the definition of $\mathcal{I}$ and the functions $H_i$, this yields that $H(x)$ and $H(x')$ agree on at least $\eta$ positions. We construct an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that uses $\mathcal{A}$ to break the weak $\eta$-near-collision resistance of $\mathcal{H}$. Note that the choice of $\mathcal{I}$ is independent of $H \in \mathcal{H}$. $\mathcal{B}_1$ therefore just encodes $\mathsf{K} = (\mathsf{K}_0, \ldots, \mathsf{K}_\ell)$ to $\mathcal{J} \subseteq \{1, \ldots, n\}$ with $|\mathcal{J}| = \eta$. $\mathcal{B}_2$ simply relays $\mathcal{A}$'s output $(X^{(1)}, \ldots, X^{(Q)}, X^*)$. The runtime $t_{\mathcal{B}}$ of $\mathcal{B}$ is at most $2t_{\mathcal{A}}$, since $\mathcal{B}$ does nothing more than executing $\mathcal{A}$ and relaying its outputs. Therefore, we get

$$\Pr[\mathsf{coll}] > \varepsilon_{\mathcal{A}}/2 \geq \frac{2t_{\mathcal{A}}(2t_{\mathcal{A}} - 1)}{2^{\eta}} \geq \frac{t_{\mathcal{B}}(t_{\mathcal{B}} - 1)}{2^{\eta+1}},$$

where the second inequality follows from Equation (3.35). This contradicts the weak near-collision resistance of $\mathcal{H}$. Next, we determine $\Pr[\neg\mathsf{badChal}]$. We have that the events $\mathsf{coll}$ and $\mathsf{badChal}$ are independent of each other because $(\mathsf{K}_0, \ldots, \mathsf{K}_\ell)$ is chosen independently from $(X^{(1)}, \ldots, X^{(Q)}, X^*)$. Moreover, each $\mathsf{K}_i$ with $i \in \mathcal{I}$ is chosen uniformly at random from $\{0,1\}^{2^i}$ and thus we have

$$\Pr[\neg\mathsf{badChal}] = \Pr[H_i(X^*) = \mathsf{K}_i \text{ for all } i \in \mathcal{I}] = \frac{1}{2^{\sum_{i \in \mathcal{I}} 2^i}} = 2^{-\eta},$$

where the last equation follows by definition of $\eta$. To prove Property 1, we then calculate

$$(\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}])2^{-\eta} \geq \left(\varepsilon_{\mathcal{A}} - \frac{\varepsilon_{\mathcal{A}}}{2}\right) \frac{\varepsilon_{\mathcal{A}}}{16t_{\mathcal{A}}^2 - 8t_{\mathcal{A}}} = \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}},$$

where the first inequality follows from Equation (3.35). Finally, to show Property 2, we explain that if $\mathsf{badEval}$ occurs, then either $\mathsf{badChal}$ or $\mathsf{coll}$ *must occur*. This is because if there exists $x \in \{X^{(1)}, \ldots, X^{(Q)}\}$ with $x \neq X^*$ and $H_i(x) = \mathsf{K}_i$ for all $i \in \mathcal{I}$, then we have either that also $H_i(X^*) = \mathsf{K}_i$ for all $i \in \mathcal{I}$ and then $\mathsf{coll}$ occurs or we have that there exists an index $i \in \mathcal{I}$ such that $H_i(X^*) \neq \mathsf{K}_i$ and then $\mathsf{badChal}$ occurs. This concludes the proof. □

Plausibility of weak near-collision resistance for standard hash functions. Near-collision resistance has been studied in several previous works, such as [BC04, BCJ+05, PS14]. Furthermore, the Handbook of Applied Cryptography [MvOV96, Remark 9.22] lists near-collision resistance as a desired property of hash functions and a potential *certificational property*. Moreover, the sponge construction for hash functions, which SHA-3 is based on, has been shown to be indifferentiable from a random oracle [BDPV08] in a slightly idealized model, which immediately implies near-collision resistance of the sponge construction in this model. Since weak near-collision resistance is an even weaker property, we view it as a natural property of cryptographic hash functions.

Near-collision resistance and the non-programmable random oracle model. Near-collision resistance holds *unconditionally* in the non-programmable random oracle model [FLR+10]. Hence, our results can also be viewed as a generic technique to obtain adaptively-secure cryptosystems in the non-programmable random oracle model without any additional assumptions. In this sense, our paper is in line with recent works that aim to avoid programmability, such as [FHJ20].

Comparison to ELFs. *Extremely lossy function*s (ELFs), which were introduced by Zhandry in [Zha16, Zha19a, Zha19b], are hash functions that allow the reductions to choose the hash function's image size depending on the adversary, such that a function with a small image size is indistinguishable from an injective hash function. Blockwise partitioning uses the weak near-collision resistance of standard hash functions in a similar manner, by selecting the blocks depending on the adversary's runtime and advantage. Hence, ELFs have the potential to enable constructions similar to the ones we present. However, the known construction [Zha19a] based on exponential hardness of the decisional Diffie-Helman problem relies on public key techniques and thus is less efficient then a standard hash functions. Blockwise partitioning can also be seen as an approach towards resolving the open problem of constructing ELFs from symmetric-key primitives. While we syntactically do not construct an ELF, the way blockwise partitioning is used in a proof is very similar.

Comparison to confined guessing. Note that the index set $\mathcal{I}$ defined above may contain *multiple* indices. This is a major difference of our approach to confined guessing and the application of truncation collision resistance in [JK18], where always only *single* blocks are guessed.

The advantage of being able to guess multiple blocks is that we are now able to define $\eta$ as fine grained as we previously did for cAHFs in Section 3.3. In contrast, [BHJ+13, BHJ+15] and [JK18] were only able to pick values $\eta$ of exponentially increasing size, such that $\eta = 2^{2^j}$ for some $j$, which is the reason why our reductions can improve tightness and the strength of the required assumptions quadratically. At the same, blockwise partitioning allows us to only consider a logarithmic number of blocks and thus enables us to construct schemes with efficiency similar to what a confined guessing approach achieves.

However, the downside of blockwise partitioning is that it does not enable black-box transformations from selective security to adaptive security as confined guessing does. Essentially, the confined guessing technique enables this by running $\mathcal{O}(\log \lambda)$ many instances of a scheme in parallel, one for each block. Since it only guesses one block, it allows to reduce the adaptive security of the scheme to the selective security of an instance for that one block. Since blockwise partitioning guesses the challenge value of several blocks instead of one, this black-box approach of [BHJ+13, BHJ+15] and [JK18] is not applicable to blockwise partitioning.

### 3.4.2 Verifiable Random Functions from Blockwise Partitioning

The construction of our VRF is an adaption of Yamada's VRF [Yam17a] to blockwise partitioning. For simplicity, we let $\ell := \lfloor \log(2\lambda + 3) \rfloor$. Moreover, note that for all $X \in \{0,1\}^*$ and all $i \in [\ell]_0$ we view $H_i(X)$ as a number in $\mathbb{Z}_p$ by interpreting it as the canonical binary representation of a natural number. This requires that $p > 2^\ell - 1$. However, since

$$2^\ell = 2^{\lfloor \log(2\lambda+3) \rfloor} \leq 2\lambda + 3$$

holds, this does not require us to use any significantly larger $p$ than we would do anyway for security. We thus proceed to formally describe the VRF.

**Construction 2.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions for some $n \in \mathbb{N}$, let $\mathsf{GrpGen}$ be a certified bilinear group generator and let $\mathcal{VRF}_{\mathsf{Blk}} = (\mathsf{SetupVRF}_{\mathsf{Blk}}, \mathsf{Eval}_{\mathsf{Blk}}, \mathsf{Vfy}_{\mathsf{Blk}})$ be the following algorithms.

**Key generation:** $\mathsf{SetupVRF}_{\mathsf{Blk}}(1^\lambda)$ chooses a group description $\mathcal{BG} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$, a random hash function $H \xleftarrow{\$} \mathcal{H}$, random generators $g, h \xleftarrow{\$} \mathbb{G}^*$ and samples $w_i \xleftarrow{\$} \mathbb{Z}_p$ for all $i \in [\ell]_0$. It then sets $W_i := g^{w_i}$ for all $i \in [\ell]$ and returns

$$\mathsf{vk} := (\mathcal{BG}, g, h, (W_i)_{i\in[\ell]_0}, H) \qquad \text{and} \qquad \mathsf{sk} := ((w_i)_{i\in[\ell]_0}).$$

**Evaluation:** $\mathsf{Eval}_{\mathsf{Blk}}(\mathsf{sk}, X)$ computes

$$\Theta_i := \prod_{i'=0}^{i} (w_{i'} + H_{i'}(X)),$$

for all $i \in [\ell]_0$. If there is an index $i \in [\ell]_0$ such that $\Theta_i(X) \equiv 0 \bmod p$, it sets $Y := 1_{\mathbb{G}_T}$ and $\pi_i = 1_{\mathbb{G}}$ for all $i \in [\ell]_0$. Otherwise, it sets

$$Y := e(g,h)^{1/\Theta_\ell} \qquad \text{and} \qquad \pi_i := g^{1/\Theta_i}$$

for all $i \in [\ell]_0$ and outputs $(Y, (\pi_i)_{i\in[\ell]_0})$.

**Verification:** $\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi)$ checks if the following conditions are met and outputs 0 if not, otherwise it outputs 1.

1. $X \in \{0,1\}^*$
2. $\mathsf{vk}$ is of the form $(\mathcal{BG}, g, h, (W_i)_{i\in[\ell]_0}, H)$.
3. $\mathcal{BG}$ is a certified encoding of a bilinear group: $\mathsf{GrpVfy}(1^\lambda, \mathcal{BG}) = 1$.
4. All group elements are correctly encoded, that is: the generators $g$ and $h$ are correctly encoded, as verified by running $\mathsf{GrpElemVfy}(1^\lambda, \mathcal{BG}, g) = 1$ and $\mathsf{GrpElemVfy}(1^\lambda, \mathcal{BG}, h) = 1$, and all group elements $W_i$ and $\pi_i$ are correctly encoded, which is verified by running $\mathsf{GrpElemVfy}(1^\lambda, \mathcal{BG}, W_i) = 1$ and $\mathsf{GrpElemVfy}(1^\lambda, \mathcal{BG}, \pi_i) = 1$ for all $i \in [\ell]_0$.

5. If there is an $i \in [\ell]_0$ such that $W_i \cdot g^{H_i(X)} = 1_{\mathbb{G}}$, then it holds that $Y = 1_{\mathbb{G}_T}$ and $\pi_i = 1_{\mathbb{G}}$ for all $i \in [\ell]_0$.
6. If $W_i \cdot g^{H_i(X)} \neq 1_{\mathbb{G}}$ for all $i \in [\ell]_0$, then it holds that $e(\pi_0, W_i \cdot g^{H_0(X)}) = 1_{\mathbb{G}_T}$ and that $e(\pi_i, W_i \cdot g^{H_i(X)}) = e(g, \pi_{i-1})$ for all $i \in [\ell]$.
7. It holds that $e(\pi_\ell, h) = Y$.

**Correctness and Unique Provability of $\mathcal{VRF}_{\mathsf{Blk}}$.**

The correctness and uniqueness of $\mathcal{VRF}_{\mathsf{Blk}}$ follow with similar arguments to those by Yamada in the full version [Yam17b].

Correctness. We prove the correctness of $\mathcal{VRF}_{\mathsf{Blk}}$ by considering an arbitrary input $X \in \{0,1\}^*$ and assume that $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}_{\mathsf{Blk}}(1^\lambda)$ and $(Y, \pi) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{Blk}}(\mathsf{sk}, X)$ are honestly generated. We have that $\mathsf{vk} = (\mathcal{BG}, g, h, (W_i)_{i\in[\ell]_0}, H)$ and $\pi = (\pi_i)_{i\in[\ell]_0}$. As can easily be verified, $(\mathsf{vk}, X, Y, \pi)$ passes the first four properties that are checked in $\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi)$. For the remaining conditions, distinguish the two cases that there exists or does not exist an index $i \in [\ell]_0$ such that $\Theta_i(X) \equiv 0 \bmod p$. If there exists such an index $i \in [\ell]_0$, then $\mathsf{Eval}_{\mathsf{Blk}}$ sets $Y := 1_{\mathbb{G}_T}$ and $\pi_i := 1_{\mathbb{G}}$ for all $i \in [\ell]_0$. Thus, conditions 5, 6 and 7 also do not lead to $\mathsf{Vfy}_{\mathsf{Blk}}$ rejecting its input and it is just accepted.

We proceed by considering the case that there does not exist an index $i \in [\ell]_0$ such that $\Theta_i(X) \equiv 0 \bmod p$. In this case we have that

$$\begin{aligned} e\left(\pi_i, W_i \cdot g^{H_i(X)}\right) &= e\left(g^{1/\Theta_i}, g^{w_i} \cdot g^{H_i(X)}\right) \\ &= e\left(g^{1/\Theta_i}, g^{w_i + H_i(X)}\right) \\ &= e\left(g, g\right)^{\frac{w_i + H_i(X)}{\Theta_i}} \\ &= e\left(g, g\right)^{1/\Theta_{i-1}} = e\left(g, g^{1/\Theta_{i-1}}\right) = e(g, \pi_{i-1}) \end{aligned}$$

holds. Thus, neither the fifth nor the sixth condition lead to $\mathsf{Vfy}_{\mathsf{Blk}}$ rejecting the input. Finally, we have that

$$e(\pi_\ell, h) = e(g^{1/\Theta_\ell}, h) = e(g, h)^{1/\Theta_\ell} = Y$$

holds and thus also the seventh condition does not lead to $\mathsf{Vfy}_{\mathsf{Blk}}$ rejecting the input. Hence, $\mathsf{Vfy}_{\mathsf{Blk}}$ always accepts if all its inputs are honestly generated.

Unique Provability. In order to show that $\mathcal{VRF}_{\mathsf{Blk}}$ has unique provability, we have to show that for every $\mathsf{vk} \in \{0,1\}^*$ and $X \in \{0,1\}^*$ there do not exist bit strings $Y, \pi, Y', \pi' \in \{0,1\}^*$ with $Y \neq Y'$ such that $\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi) = \mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y', \pi') = 1$ holds. We prove this by considering arbitrary bit strings $\mathsf{vk}, X \in \{0,1\}^*$ and $Y, \pi, Y', \pi' \in \{0,1\}^*$ with $Y \neq Y'$ such that $\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi) = \mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y', \pi') = 1$ holds. We will show that this can only hold if $Y = Y'$ holds.

Recall that $\mathsf{Vfy}_{\mathsf{Blk}}$ first verifies that $X \in \{0,1\}^*$, the bilinear group is indeed certified and that all supposed group elements are correctly and uniquely encoded. We therefore assume for the remainder of the proof that these conditions hold because otherwise $\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi) = \mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y', \pi') = 1$ could not hold.

We first consider the case that there is $i \in [\ell]_0$ such that $W_i \cdot g^{H_i(X)} = 1_{\mathbb{G}}$ holds. Then, in order for $\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi) = \mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y', \pi') = 1$ to hold, both $(Y, \pi)$ and $(Y', \pi')$ have to pass Step 5 of the verification algorithm. However, they can only pass Step 5 in this case if $Y = Y' = 1_{\mathbb{G}_T}$ holds, which proves our claim in this case.

Now, we consider the case that there is no index $i \in [\ell]_0$ such that $W_i \cdot g^{H_i(X)} = 1_{\mathbb{G}}$ holds. Then, in order to pass Step 6 of the verification algorithm, it has to hold that $e(\pi_0, W_0 \cdot g^{H_0(X)}) = 1_{\mathbb{G}_T} = e(\pi'_0, W_0 \cdot g^{H_0(X)})$. The group structure together with the unique representation guaranteed by the certified bilinear group generator thus implies that $\pi_0 = \pi'_0$ has to hold. Furthermore, the sixth condition in the verification algorithm requires that

$$e(\pi_i, W_i \cdot g^{H_i(X)}) = e(g, \pi_{i-1}) \qquad \text{and} \qquad e(\pi'_i, W_i \cdot g^{H_i(X)}) = e(g, \pi'_{i-1})$$

has to hold for all $i \in [\ell]$. By induction, the group structure together with the unique representation guaranteed by the certified bilinear group generator thus implies that $\pi_i = \pi'_i$ has to hold for all $i \in [\ell]$ if

$$\mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y, \pi) = \mathsf{Vfy}_{\mathsf{Blk}}(\mathsf{vk}, X, Y', \pi') = 1$$

holds. Finally, Step 7 of the verification algorithm enforces that

$$e(\pi_\ell, h) = Y \qquad \text{and} \qquad e(\pi'_\ell, h) = Y'$$

holds. Since we already established that $\pi_\ell = \pi'_\ell$ holds, it also has to hold that

$$Y = Y'$$

by the group structure and the unique representation guaranteed by the certified bilinear group generator. This proves our claim also for the second case and by that the unique provability of $\mathcal{VRF}_{\mathsf{Blk}}$.

### Pseudorandomness of $\mathcal{VRF}_{\mathsf{Blk}}$.

We prove the security of $\mathcal{VRF}_{\mathsf{Blk}}$ from the $q$-DBDHI assumption (see Definition 14).

**Theorem 6.** *If $\mathcal{VRF}_{\mathsf{Blk}}$ is instantiated with a family $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^{2\lambda+3}\}$ of weak near-collision resistant hash functions (see Definition 21), then for any legitimate attacker $\mathcal{A}$ that breaks the pseudorandomness of $\mathcal{VRF}_{\mathsf{Blk}}$ in time $t_{\mathcal{A}}$ with advantage $\varepsilon_{\mathcal{A}} := \mathsf{Adv}^{\mathsf{Psrd}}_{\mathcal{VRF}_{\mathsf{Blk}},\mathcal{A}}(\lambda)$ let $\mathsf{K} \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ and let $\mathcal{I} := \{i \in [\ell]_0 : \mathsf{K}_i \neq \perp\}$. Then there exists an algorithm $\mathcal{B}$ that, given (sufficiently close approximations of) $t_{\mathcal{A}}$ and $\varepsilon_{\mathcal{A}}$, breaks the $q$-DBDHI assumption with*

$$q := |\mathcal{I}| + 2\sum_{i \in \mathcal{I}} \left(2^{2^i} - 1\right) \tag{3.36}$$

*in time* $t_{\mathcal{B}} \approx t_{\mathcal{A}}$ *and with*

$$\mathsf{Adv}_{\mathcal{B}}^{q\text{-DBDHI}}(\lambda) \geq \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}} + \mathsf{negl}(\lambda),$$

*for some negligible function* $\mathsf{negl}(\cdot)$*. In particular, if* $\mathcal{VRF}_{\mathsf{Blk}}$ *is instantiated with a bilinear group generator for which the* $q$-DBDHI *assumption is hard, then* $\mathcal{VRF}_{\mathsf{Blk}}$ *is a secure verifiable random function as defined in Definition 6.*

Before we proceed with the proof of Theorem 6, we succinctly discuss the size of $q$ in the $q$-type assumption. First, observe that even though it depends on $\mathsf{K} \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$, which is randomized, $\mathcal{I}$ and $|\mathcal{I}|$ in particular are deterministically defined by $t_{\mathcal{A}}$ and $\varepsilon_{\mathcal{A}}$. Thus, $q$ is well defined. Furthermore, the sum in Equation (3.36) can easily be upper bounded as we observe in the following lemma, which will also use in the proof of Theorem 6 below.

**Lemma 7.** *For arbitrary* $t \in \mathbb{N}$ *and* $\varepsilon \in [0,1]$ *such that* $t/\varepsilon \leq 2^\lambda$ *let* $\mathsf{K} \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t, \varepsilon)$ *and let the index set be* $\mathcal{I} \coloneqq \{i : K_i \neq \bot\} \subseteq [\lfloor \log(2\lambda + 3) \rfloor]_0$*, it then holds that*

$$2 \cdot \sum_{i \in \mathcal{I}} \left(2^{2^i} - 1\right) \leq \frac{32t^2}{\varepsilon}.$$

*Proof.*

$$\begin{aligned} 2 \cdot \sum_{i \in \mathcal{I}} \left(2^{(2^i)} - 1\right) &\leq 2 \cdot \sum_{i \in \mathcal{I}} 2^{(2^i)} < 2 \cdot \prod_{i \in \mathcal{I}} 2^{(2^i)} \\ &= 2 \cdot 2^{\sum_{i \in \mathcal{I}} (2^i)} = 2^\eta \\ &= 2 \cdot 2^{\lceil \log(4t(2t-1)/\varepsilon) \rceil} \leq \frac{8t(2t-1)}{\varepsilon} \\ &\leq 2 \cdot \frac{16t^2}{\varepsilon} = \frac{32t^2}{\varepsilon}, \end{aligned} \tag{3.37}$$

where the second inequality in Equation (3.37) holds, because $a + b \leq ab$ for all $a, b \geq 2$. This concludes the proof. □

Moreover, we will use the *Schwartz-Zippel Lemma* in the proof of Theorem 6. We therefore state it here for reference.

**Lemma 8** (Schwartz-Zippel lemma [Zip79, Sch80])**.** *Let* $f \in \mathbb{F}[\mathsf{Z}]$ *be a non-zero polynomial of degree* $d \geq 0$ *over the field* $\mathbb{F}$*. For any finite subset* $S$ *of the field* $\mathbb{F}$ *and for any* $r_1, \ldots, r_n$ *drawn uniformly and independently at random from* $S$ *it holds that*

$$\Pr\left[f(r_i) = 0 \,\forall\, i \in [n]\right] \leq \frac{d}{|S|}.$$

With this preparation, we now proceed with the proof of Theorem 6.

*Proof of Theorem 6.* We prove Theorem 6 with a sequence of games argument (see Section 2.5). We denote the event that Game $i$ outputs 1 by $\mathsf{G}_i$ and denote the adversary's advantage in Game $i$ by $\mathsf{E}_i := |\Pr[\mathsf{G}_i] - 1/2|$. The first half of the proof closely follows the proofs by Jager, Kurek and Niehues [JK18, JN19a]. The second half follows the proof by Yamada [Yam17a].

*Game 0.* This is the original VRF pseudorandomness experiment $G^{\mathsf{Psrd}}$ from Definition 6. And therefore

$$\Pr[\mathsf{G}_0] = \Pr\left[G^{\mathsf{Psrd}}_{\mathcal{VRF}_{\mathsf{Blk}},\mathcal{A}}(\lambda) = 1\right] \qquad \text{and} \qquad \varepsilon_{\mathcal{A}} := \mathsf{E}_0 = \left|\mathsf{G}_0 - \frac{1}{2}\right|.$$

*Game 1.* This game is identical to Game 0, except that the challenger runs $\mathsf{K} = (\mathsf{K}_0, \ldots, \mathsf{K}_\ell) \xleftarrow{\$} \mathsf{BPSmp}(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ depicted in Figure 3.1. Furthermore, it defines $\mathcal{I} := \{i : \mathsf{K}_i \neq \bot\}$. Moreover, we let $\mathcal{X}$ be the set of all queries that the adversary makes to $\mathsf{Eval}_{\mathsf{Blk}}(\mathsf{sk}, X^{(i)})$, and let $\mathcal{X}^* := \mathcal{X} \cup \{X^*\}$, where $X^*$ is the challenge query. Then note that the choice of $\mathsf{K}$ also defines the events $\mathsf{coll}, \mathsf{badChal}$ and $\mathsf{badEval}$ as in Lemma 6 as

$$\begin{aligned}
\mathsf{badChal} &\iff \exists\, i \in \mathcal{I} : H_i(X^*) = \mathsf{K}_i, \\
\mathsf{coll} &\iff \exists\, X \neq X' \in \mathcal{X}^* \text{ s.t. } H_i(X) = H_i(X')\, \forall\, i \in \mathcal{I} \text{ and} \\
\mathsf{badEval} &\iff x \in \mathcal{X} \text{ s.t. } H_i(X) = \mathsf{K}_i\, \forall\, i \in \mathcal{I}.
\end{aligned}$$

Since these changes are purely conceptual, it holds that

$$\Pr[\mathsf{G}_1] = \Pr[\mathsf{G}_0].$$

*Game 2.* In this game, the challenger aborts the experiment and outputs a random bit if it detects that event $\mathsf{coll}$ occurred. We thus have that

$$\begin{aligned}
\Pr[\mathsf{coll}] &\geq |\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| && (3.38)\\
&= \left|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2] + \frac{1}{2} - \frac{1}{2}\right| \\
&= \left|\left(\Pr[\mathsf{G}_1] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| \\
&= \left|\left(\Pr[\mathsf{G}_1] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| + \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| - \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| \\
&\geq \left|\left(\Pr[\mathsf{G}_1] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right) + \left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right)\right| - \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| && (3.39)\\
&= \left|\Pr[\mathsf{G}_1] - \frac{1}{2}\right| - \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| = \mathsf{E}_1 - \mathsf{E}_2,
\end{aligned}$$

where Equation (3.38) follows from Lemma 1 (Shoup's Difference Lemma) and Equation (3.39) follows from the triangle inequality. Rearranging the terms above, we thus obtain that

$$\mathsf{E}_2 \geq \mathsf{E}_1 - \Pr[\mathsf{coll}] = \varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]$$

holds.

*Game 3.* In this game, the challenger additionally aborts and outputs a random bit if badChal or badEval occur. By Property 2 of Lemma 6 we have $\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal}$. Thus, event badEval does not cause any aborts on its own. Moreover we have by Property 1 from Lemma 6 that the events badChal and coll are independent and that

$$(\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]) \cdot \Pr[\neg\mathsf{badChal}] \geq \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}}$$

holds. We thus conclude that

$$\mathsf{E}_3 = \mathsf{E}_2 \cdot \Pr[\neg\mathsf{badChal}] = (\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]) \cdot \Pr[\neg\mathsf{badChal}] \geq \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}} \tag{3.40}$$

holds.

*Game 4.* In this game, we replace the events badChal, badEval and coll by a single event $\mathsf{bad} := \mathsf{badChal} \vee \mathsf{badEval} \vee \mathsf{coll}$ and the challenger outputs a random bit if bad occurs. Since this is a purely conceptual change, it does not affect $\mathcal{A}$'s success probability and we thus have that

$$\mathsf{E}_4 = \mathsf{E}_3.$$

*Game 5.* In this game, the challenger aborts as soon as the event bad occurs instead of in the end. Since Game 5 and Game 4 are *identical until* bad we have that

$$\mathsf{E}_5 = \mathsf{E}_4.$$

The previous changes were about applying the blockwise partitioning technique from Lemma 6. The next changes use Yamada's technique to embed the partitioning into the public key and therefore closely follow his proof [Yam17a]. Recall that we set

$$q := j + 2\sum_{i \in \mathcal{I}} \left(2^{2^i} - 1\right),$$

where for $j := |\mathcal{I}|$.

*Game 6.* In this game we change the way $\mathsf{vk}$ is generated by the challenger. It samples $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, $\tilde{w}_i \xleftarrow{\$} \mathbb{Z}_p^*$ and sets

$$w_i := \begin{cases} \tilde{w}_i \cdot \alpha - \mathsf{K}_i & \text{if } \mathsf{K}_i \neq \perp \\ \tilde{w}_i & \text{otherwise} \end{cases}$$

for all $i \in [n]_0$. If $w_i = 0$ for any $i \in [\ell]_0$, then the challenger aborts and outputs a random bit. This happens *iff* $\tilde{w}_i \alpha = \mathsf{K}_i$. Since $\tilde{w}_i \alpha$ is distributed uniformly at random in $\mathbb{Z}_p^*$, the probability that this happens for any of the $\mathcal{O}(\log \lambda)$ many $w_i$ is negligible and therefore

$$|\mathsf{E}_6 - \mathsf{E}_5| \leq \mathsf{negl}(\lambda).$$

*Game 7.* In this game we change the way the challenger chooses the generator $g$. Again, let $j := |\mathcal{I}|$ be the number positions in $\mathsf{K}$ that are not $\perp$. The challenger then first defines the polynomial $\mathsf{Q}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ as

$$\mathsf{Q}(\mathsf{Z}) := \mathsf{Z}^{j-1} \prod_{i \in \mathcal{I}} \prod_{\substack{-2^{2^i}+1 \leq k \leq 2^{2^i}-1 \\ k \neq 0}} (\tilde{w}_i \mathsf{Z} + k)\,. \tag{3.41}$$

Now, instead of sampling $g \xleftarrow{\$} \mathbb{G}$, the challenger samples $\tilde{g} \xleftarrow{\$} \mathbb{G}$ and sets $g := \tilde{g}^{\mathsf{Q}(\alpha)}$. If $g = 1_{\mathbb{G}}$, which happens *iff* $\mathsf{Q}(\alpha) \equiv 0 \bmod p$, the challenger outputs a random bit and aborts. It can be seen that the distribution of $g$ changes only if $\mathsf{Q}(\alpha) \equiv 0 \bmod p$. Since $\mathsf{Q}(\mathsf{Z})$ is a non-zero polynomial of degree $j - 1 + 2\sum_{i \in \mathcal{I}}^{\ell}(2^{(2^i)} - 1) = q - 1 \leq \lfloor \log(2\lambda + 3) \rfloor + 32t_{\mathcal{A}}^2/\varepsilon$ by Lemma 7 and $\alpha$ is uniformly random in $\mathbb{Z}_p^*$, we have that $\Pr[\mathsf{Q}(\alpha) = 0] \leq \frac{\ell + 32t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}}}{p-1}$ holds by the Schwartz-Zippel lemma (see Lemma 8). Since $\lfloor \log(2\lambda + 3) \rfloor + 32t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}}$ is polynomial in $\lambda$ and $p \in 2^{\Omega(\lambda)}$ by the properties of $\mathsf{GrpGen}$, it holds that

$$|\mathsf{E}_7 - \mathsf{E}_6| = \mathsf{negl}(\lambda).$$

*Game 8.* In this game, the challenger always responds to $\mathcal{A}$'s challenge with a random element $Y \xleftarrow{\$} \mathbb{G}_T$, regardless of the value of $b$. Thus, it holds that

$$\Pr[\mathsf{G}_8 = 1] = \frac{1}{2} \qquad \text{and} \qquad \mathsf{E}_8 = 0.$$

Furthermore, we claim that there is an algorithm $\mathcal{B}$ such that

$$|\mathsf{E}_8 - \mathsf{E}_7| = \mathsf{Adv}_{\mathcal{B}}^{q\text{-}\mathsf{DBDHI}}(\lambda).$$

We proceed by describing how $\mathcal{B}$ works.

INITIALIZATION. In the beginning of the experiment $\mathcal{B}$ runs $\mathsf{K} := (\mathsf{K}_0, \ldots, \mathsf{K}_\ell) \xleftarrow{\$} \mathsf{BPSmp}(t_\mathcal{A}, \varepsilon_\mathcal{A})$ and by doing so determines the index set $\mathcal{I} = \{i : \mathsf{K}_i \neq \bot\}$ according to Lemma 6. Moreover, it receives a $q$-DBDHI instance $(\mathcal{BG}, \tilde{g}, h, \tilde{g}^\alpha, \tilde{g}^{\alpha^2}, \ldots, \tilde{g}^{\alpha^q}, T)$, for $q = j + \sum_{i\in\mathcal{I}} \left(2^{2^i} - 1\right)$, where either $T = e(\tilde{g}, h)^{1/\alpha}$ or $T \xleftarrow{\$} \mathbb{G}_T$. Recall that $j = |\mathcal{I}|$ is the number of non-wildcard positions in $\mathsf{K}$. In order to compute $g$, $\mathcal{B}$ samples $\tilde{w}_i \xleftarrow{\$} \mathbb{Z}_p^*$ for all $i \in [\ell]_0$ as in Game 8. It then computes coefficients $\varphi_0, \ldots \varphi_{q-1} \in \mathbb{Z}_p$ such that

$$\mathsf{Q}(\mathsf{Z}) = \mathsf{Z}^{j-1} \prod_{i\in\mathcal{I}}^{\ell} \prod_{\substack{-2^{2^i}+1 \leq k \leq 2^{2^i}-1 \\ k\neq 0}} (\tilde{w}_i \mathsf{Z} + k) = \sum_{k=0}^{q-1} \varphi_k \mathsf{Z}^k$$

holds, where $\mathsf{Q}(\mathsf{Z})$ is as in Equation (3.41). Such coefficients $\varphi_0, \ldots \varphi_{j\cdot 2(a-1)+j-1} \in \mathbb{Z}_p$ exist because $\mathsf{Q}(\mathsf{Z})$ is of degree $q-1$ and $\mathcal{B}$ can compute them since it knows $\tilde{w}_i$ for all $i \in [n]_0$. It then computes

$$g := \prod_{k=0}^{q-1} \left(\tilde{g}^{\alpha^k}\right)^{\varphi_k} = \tilde{g}^{\sum_{k=0}^{q-1} \varphi_k \alpha^k} = \tilde{g}^{\mathsf{Q}(\alpha)}.$$

Recall that $g^{(\alpha^0)}, \ldots, g^{(\alpha^{q-1})}$ are given in the $q$-DBDHI instance and $\mathcal{B}$ can therefore compute $g$ in this way. If $g = 1_\mathbb{G}$, meaning $\mathsf{Q}(\alpha) \equiv 0 \bmod p$, $\mathcal{B}$ aborts and outputs a random bit. Since $\alpha$ from the $q$-DBDHI instance is distributed identically to $\alpha$ in Game 8 and all $w_i$ are distributed exactly as in Game 8, we have that $g$ is distributed exactly as in Game 8. $\mathcal{B}$ proceeds with computing $\mathsf{vk}$. For this purpose, $\mathcal{B}$ (implicitly) sets

$$w_i := \begin{cases} \tilde{w}_i \cdot \alpha - \mathsf{K}_i & \text{if } \mathsf{K}_i \neq \bot \\ \tilde{w}_i & \text{otherwise} \end{cases}$$

by computing $\psi_{i,0}, \ldots, \psi_q \in \mathbb{Z}_p$ for all $i \in [\ell]_0$ such that

$$\mathsf{Q}(\mathsf{Z}) \cdot (\tilde{w}_i \cdot \mathsf{Z} - \mathsf{K}_i) = \sum_{k=0}^{q} \psi_{i,k} \mathsf{Z}^k$$

holds and setting

$$W_i := \prod_{k=0}^{q} \left(\tilde{g}^{\alpha^k}\right)^{\psi_{i,k}} = \tilde{g}^{\sum_{k=0}^{q} \mathsf{Q}(\alpha)\cdot(\tilde{w}_i\alpha - \mathsf{K}_i)} = g^{w_i}$$

for all $i \in [\ell]_0$ with $\mathsf{K}_i \neq \bot$. Observe that $\mathcal{B}$ can compute these values by using $\tilde{g}^{(\alpha^0)}, \ldots, \tilde{g}^{(\alpha^q)}$ from the $q$-DBDHI instance. For all $i \in [\ell]$ with $\mathsf{K}_i = \bot$, $\mathcal{B}$ then sets $W_i := g^{w_i}$ and then gives $\mathsf{vk} := (\mathcal{BG}, g, h, (W_i)_{i\in[\ell]_0}, H)$ to $\mathcal{A}$. Further, since $\alpha$ and all $w_i$ are distributed as in Game 8, we have that all $W_i$ are also distributed as in Game 8.

HELPFUL DEFINITIONS. In order to describe how $\mathcal{B}$ responds to evaluation queries and the challenge $X^*$, we define the polynomials $\mathsf{P}_{X,i}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ — these polynomial will assume the role of the respective $\Theta_i$ in the construction — for all $X \in \{0,1\}^*$ and $i \in [\ell]_0$ as

$$\mathsf{P}_{X,i}(\mathsf{Z}) := \begin{cases} \mathsf{P}_{X,i-1}(\mathsf{Z})\,(\tilde{w}_i\mathsf{Z} - \mathsf{K}_i + H_i(X)) & \text{if } \mathsf{K}_i \neq \bot \\ \mathsf{P}_{X,i-1}(\mathsf{Z})\,(\tilde{w}_i + H_i(X)) & \text{if } \mathsf{K}_i = \bot \end{cases},$$

where $\mathsf{P}_{X,-1}(\mathsf{Z}) := 1$. Furthermore, we let $\mathsf{P}_X(\mathsf{Z}) := \mathsf{P}_{X,\ell}(\mathsf{Z})$. We require the following lemma by Yamada [Yam17a] to proceed with the description of $\mathcal{B}$.

**Lemma 9** (Lemma 6 in [Yam17a])**.** *Let $X \in \{0,1\}^*$ then there exist $\zeta_X \in \mathbb{Z}_p^*$ and $\mathsf{R}_X(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ such that*

$$\frac{\mathsf{Q}(\mathsf{Z})}{\mathsf{P}_X(\mathsf{Z})} = \begin{cases} \frac{\zeta_X}{\mathsf{Z}} + \mathsf{R}_X(\mathsf{Z}) & \textit{if } H_i(X) = \mathsf{K}_i \textit{ for all } i \in \mathcal{I} \textit{ and} \\ \mathsf{R}_X(\mathsf{Z}) & \textit{otherwise.} \end{cases}$$

The proof of Lemma 9 mostly follows the proof of Lemma 6 in [Yam17a]. However, the proof is only contained in the full version [Yam17b]. As to not interrupt the current proof, we provide it after finishing this proof.

ANSWERING EVALUATION QUERIES. When $\mathcal{B}$ receives an evaluation query $X \in \{0,1\}^*$ from $\mathcal{A}$, it checks whether $H_i(X) = \mathsf{K}_i$ for all $i \in \mathcal{I}$ and if so aborts and outputs a random bit. If there is an index $i \in \mathcal{I}$ such that $H_i(X) \neq \mathsf{K}_i$, then $\mathcal{B}$ lets $\mathsf{R}_{X,i}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ such that $\mathsf{R}_{X,i}(\mathsf{Z}) = \mathsf{Q}(\mathsf{Z})/\mathsf{P}_{X,i}(\mathsf{Z})$ for all $i \in [\ell]$, which is guaranteed to exist by Lemma 9. $\mathcal{B}$ then computes the coefficients $\rho_{X,i,k} \in \mathbb{Z}_p$ of the polynomials $\mathsf{R}_{X,i}(\mathsf{Z})$ for all $i \in [\ell]_0$ and $k \in [d_{X,i}]$, where $d_{X,i} \leq q$ is the degree of $\mathsf{R}_{X,i}(\mathsf{Z})$, such that $\mathsf{R}_{X,i}(\mathsf{Z}) = \sum_{k=0}^{d_{X,i}} \mathsf{Z}^k \rho_{X,i}$ for all $i \in [\ell]_0$. $\mathcal{B}$ then computes $\pi_i$ as

$$\pi_i := \prod_{k=0}^{d_{X,i}} \left(\tilde{g}^{\alpha^k}\right)^{\rho_{X,i,k}} = \tilde{g}^{\sum_{k=0}^{d_{X,i}} \alpha^k \rho_{X,i,k}} = \tilde{g}^{\mathsf{R}_{X,i}(\mathsf{Z})} = \tilde{g}^{\mathsf{Q}(\alpha)/\mathsf{P}_{X,i}(\alpha)} = g^{1/\mathsf{P}_{X,i}(\alpha)},$$

for all $i \in [\ell]_0$. It then computes the result $Y$ as

$$Y := e(\pi_n, h) = e(g,h)^{1/\mathsf{P}_X(\alpha)}$$

and outputs $(Y, (\pi_i)_{i\in[\ell]_0})$ to $\mathcal{A}$. Observe that both $Y$ and $(\pi_i)_{i\in[\ell]_0}$ are distributed exactly as in Game 8.

ANSWERING $\mathcal{A}$'S CHALLENGE. When $\mathcal{B}$ receives the challenge $X^* \in \{0,1\}^*$, it checks whether there exists an index $i \in \mathcal{I}$ such that $H_i(X) \neq \mathsf{K}_i$ and if so aborts and outputs a random bit. This behavior is identical to the challenger's behavior in Game 8. Otherwise, $\mathcal{B}$ lets $b \xleftarrow{\$} \{0,1\}$ and responds to $\mathcal{A}$ with $Y \xleftarrow{\$} \mathbb{G}_T$ if $b = 1$. If $b = 0$, then $\mathcal{B}$ lets $\zeta_{X^*} \in \mathbb{Z}_p$ and $\mathsf{R}_{X^*}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ be such that $\mathsf{Q}(\mathsf{Z})/\mathsf{P}_{X^*}(\mathsf{Z}) =$

$\zeta_{X^*}/\mathsf{Z}+\mathsf{R}_{X^*}(\mathsf{Z})$ as guaranteed by Lemma 9. $\mathcal{B}$ then computes coefficients $\gamma_{X^*,k} \in \mathbb{Z}_p$ of $\mathsf{R}_X \in \mathbb{Z}_p[\mathsf{Z}]$ for all $k \in [d_{X^*}]$, where $d_{X^*} \leq q$ is the degree of $\mathsf{R}_{X^*}(\mathsf{Z})$, such that $\sum_{k=0}^{d_{X^*}} \mathsf{Z}^k \gamma_{X^*,k}$ holds. $\mathcal{B}$ then computes the response $Y$ as

$$Y := T^{\zeta_{X^*}} e\left(\prod_{k=0}^{d_{X^*}} \left(\tilde{g}^{\alpha^k}\right)^{\gamma_{X^*,k}}, h\right) = T^{\zeta_{X^*}} e\left(\tilde{g}^{\sum_{k=0}^{d_{X^*}} \alpha^k \gamma_{X^*,k}}\right) = T^{\zeta_{X^*}} e\left(\tilde{g}^{\mathsf{R}_{X^*}(\mathsf{Z})}\right) \tag{3.42}$$

and returns $Y$ to $\mathcal{A}$.

Solving $q$-DBDHI. Finally, when $\mathcal{A}$ outputs its guess $b'$ to $\mathcal{B}$, then $\mathcal{B}$ outputs 1 as the solution to the $q$-DBDHI instance and 0 otherwise.

Analysis of $\mathcal{B}$. Recapping the construction of $\mathcal{B}$, we observe that $\mathsf{vk}$ and all answers of $\mathcal{B}$ to evaluation queries by $\mathcal{A}$ are distributed identically to the challenger's interactions with $\mathcal{A}$ in Game 8. Analyzing $\mathcal{B}$'s response to the challenge $X^*$, we distinguish the following two cases depending on the $q$-DBDHI instance.

$T = e(\tilde{g}, h)^{1/\alpha}$: In this case we have to distinguish between $b = 1$ and $b = 0$. In the latter case, we have that the following by Equation (3.42):

$$Y = e\left(\tilde{g}, h\right)^{\frac{\zeta_{X^*}}{\mathsf{Z}}} e(\tilde{g}, h)^{\mathsf{R}_{X^*}(\mathsf{Z})} = e(\tilde{g}, h)^{\frac{\zeta_{X^*}}{\mathsf{Z}}+\mathsf{R}_{X^*}(\mathsf{Z})} = e(g, h)^{1/P_{X^*}(\mathsf{Z})}$$

Since $b \xleftarrow{\$} \{0, 1\}$, this is the case with probability $1/2$. Analogously, if $b = 1$, then $\mathcal{B}$ sets $Y \xleftarrow{\$} \mathbb{G}_T$.

$T \xleftarrow{\$} \mathbb{G}_T$: In this case we have that $Y = T^{\zeta_{X^*}} e\left(\tilde{g}^{\mathsf{R}_{X^*}(\mathsf{Z})}, h\right)$ is uniformly random in $\mathbb{G}_T$, regardless of what $b$ is.

We observe that $Y$ is distributed exactly as in Game 7 if $T = e(\tilde{g}, h)^{1/\alpha}$ and that $Y$ is distributed exactly as in Game 8 if $T \xleftarrow{\$} \mathbb{G}_T$. This proves that

$$\mathsf{Adv}_{\mathcal{B}}^{q\text{-DBDHI}}(\lambda) = |\mathsf{E}_8 - \mathsf{E}_7| \tag{3.43}$$

holds. Furthermore, the running time $t_{\mathcal{B}}$ of $\mathcal{B}$ consists of the time needed to execute $\mathcal{A}$ plus the time required to respond to evaluate queries by $\mathcal{A}$. Since we included the runtime of the challenger in the runtime of the adversary in Definition 6, we have that $t_{\mathcal{B}} \approx t_{\mathcal{A}}$. To complete the proof of Theorem 6, we now only need to lower bound $\mathsf{Adv}_{\mathcal{B}}^{q\text{-DBDHI}}(\lambda)$. We have for a negligible function $\mathsf{negl} : \mathbb{N} \to [0, 1]$ that

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}}^{q\text{-DBDHI}}(\lambda) &= |\mathsf{E}_8 - \mathsf{E}_7| && (3.44)\\
&\geq |\mathsf{E}_8 - \mathsf{E}_3| + \mathsf{negl}(\lambda) && (3.45)\\
&= \mathsf{E}_3 + \mathsf{negl}(\lambda) && (3.46)\\
&\geq \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}} + \mathsf{negl}(\lambda), && (3.47)
\end{aligned}$$

Equation (3.44) follows from Equation (3.43), Equation (3.45) holds because the changes between Game 3 and Game 7 are all conceptual or negligible, Equation (3.46) holds because $\mathsf{E}_8 = 0$ and Equation (3.47) follows from Equation (3.40). This completes the proof. □

*Proof of Lemma 9.* Recall that we set $\ell := \lfloor \log(2\lambda + 3) \rfloor$, denote the set of indices $i$ with $\mathsf{K}_i \neq \perp$ by $\mathcal{I} := \{\mathsf{K}_i \neq \perp\}$ and denote the number of non-wildcard positions in $\mathsf{K}$ by $j := |\mathcal{I}|$. The proof mostly follows the respective proof by Yamada [Yam17b].

In order to decomposition $\mathsf{P}_X(\mathsf{Z})$ and $\mathsf{Q}(\mathsf{Z})$, we define

$$t_X := \prod_{i \in ([\ell]_0 \setminus \mathcal{I})} (\tilde{w}_i + H_i(X))$$

and the polynomials $\mathsf{S}_{X,i}(\mathsf{Z})$ for all $i \in \mathcal{I}$ and all $X \in \{0,1\}^*$, and the polynomials $\mathsf{Q}_i(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ for all $i \in [\ell]_0$ as

$$\mathsf{S}_{X,i}(\mathsf{Z}) := \tilde{w}_i \mathsf{Z} - \mathsf{K}_i + H_i(X) \qquad \text{and} \qquad \mathsf{Q}_i(\mathsf{Z}) = \prod_{\substack{-2^{2^i}+1 \leq k \leq 2^{2^i}-1 \\ k \neq 0}} (\tilde{w}_i \mathsf{Z} + k)\,.$$

By the definitions of $\mathsf{P}_X(\mathsf{Z})$ and $\mathsf{Q}(\mathsf{Z})$, we then have that

$$\mathsf{P}_X(\mathsf{Z}) = t_X \prod_{i \in \mathcal{I}} \mathsf{S}_{X,i}(\mathsf{Z}) \qquad \text{and} \qquad \mathsf{Q}(\mathsf{Z}) = \mathsf{Z}^{j-1} \prod_{i=0}^{\ell} \mathsf{Q}_i(\mathsf{Z})$$

holds. Now, define the set of indices $i$ where $\mathsf{K}_i$ and $H_i(X)$ match as $\mathcal{M} := \{i \in \mathcal{I} : H_i(X) = \mathsf{K}_i\}$. We first observe that $\mathsf{K}_i = H_i(X)$ for all $i \in \mathcal{I}$ *if and only if* $\mathcal{M} = \mathcal{I}$. Furthermore, we observe that for all $i \in \mathcal{M}$ it holds that $\mathsf{S}_{X,i}(\mathsf{Z}) = \tilde{w}_i \mathsf{Z} - \mathsf{K}_i + H_i(X) = \tilde{w}_i \mathsf{Z}$. Therefore, it follows from the definition of $\mathsf{Q}_i(\mathsf{Z})$ that

$$\mathsf{S}_{X,i}(\mathsf{Z}) \nmid \mathsf{Q}_m(\mathsf{Z}) \text{ for all } i \in \mathcal{M} \text{ and } m \in [\ell]_0, \tag{3.48}$$

$$\mathsf{S}_{X,i}(\mathsf{Z}) \nmid \mathsf{Z} \text{ for all } i \in \mathcal{M}. \tag{3.49}$$

We distinguish the two cases considered in Lemma 9.

Case 1: $\mathcal{I} = \mathcal{M}$. In this case the lemma claims that there is $\zeta_X \in \mathbb{Z}_p$ and $\mathsf{R}_X(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ such that

$$\frac{\mathsf{Q}(\mathsf{Z})}{\mathsf{P}_X(\mathsf{Z})} = \frac{\zeta_X}{\mathsf{Z}} + \mathsf{R}_X(\mathsf{Z})$$

holds. Since $\mathcal{M} = \mathcal{I}$, we have by Equation (3.48) and by Equation (3.49) that

$$\mathsf{P}_X(\mathsf{Z}) \mid \mathsf{Z}^j \prod_{i=0}^{\ell} \mathsf{Q}_i(\mathsf{Z}) \qquad \text{but} \qquad \mathsf{P}_X(\mathsf{Z}) \nmid \mathsf{Z}^{j-1} \prod_{i=0}^{\ell} \mathsf{Q}_i(\mathsf{Z}) = \mathsf{Q}(\mathsf{Z})$$

holds, because $j = |\mathcal{I}|$. This implies the existence of a polynomial $\mathsf{R}_X(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ and $\zeta_X \in \mathbb{Z}_p$ such that $\mathsf{Q}(\mathsf{Z})/\mathsf{P}_X(\mathsf{Z}) = \zeta_X/\mathsf{Z} + \mathsf{R}_X(\mathsf{Z})$ as claimed by the lemma.

Case 2: $\mathcal{I} \neq \mathcal{M}$. In this case there exists $i^* \in \mathcal{I}$ such that $H(X)_{i^*} \neq \mathsf{K}_{i^*}$. Let $L := -\mathsf{K}_{i^*} + H_{i^*}(X)$. We then note that $L \neq 0$ and $-2^{2^{i^*}} + 1 \leq L \leq 2^{2^{i^*}} - 1$ holds. Furthermore, it holds that $\mathsf{S}_{X,i^*}(\mathsf{Z}) = \tilde{w}_{i^*}\mathsf{Z} - \mathsf{K}_{i^*} + H_{i^*}(X) = \tilde{w}_{i^*}\mathsf{Z} + L$ by the definition of $\mathsf{S}_{X,i^*}(\mathsf{Z})$. In particular, it therefore holds that $\mathsf{S}_{X,i^*}(\mathsf{Z}) \mid \mathsf{Q}_{i^*}(\mathsf{Z})$. Therefore, we also have that $\mathsf{P}_X(\mathsf{Z}) \mid \mathsf{Q}(\mathsf{Z})$ and hence it exists a polynomial $\mathsf{R}_X(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ such that $\mathsf{R}_X(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}] = \mathsf{Q}(\mathsf{Z})/\mathsf{P}_X(\mathsf{Z})$ as claimed by the lemma. This completes the proof Lemma 9. □

## 3.5 Comparison of VRF Instantiations

In this section, we use the bounds we introduced in Section 3.2 to compare the concrete number of group-/$\mathbb{Z}_p$-elements in keys and proofs of $\mathcal{VRF}_{\mathsf{cAHF}}$ in Construction 1, $\mathcal{VRF}_{\mathsf{Blk}}$ in Construction 2 and the VRFs by Jager [Jag15], Yamada [Yam17a] and Katsumata [Kat17] in detail. Furthermore, we discuss the VRF of Kohl [Koh19], which achieves very short proofs while relying on standard assumptions. Furthermore, where possible, we compare an instantiation with bAHFs from ECCs with an instantiation with cAHFs from TCR hash functions.

Except for Construction 2, all these VRFs can be instantiated in two different ways: using bAHFs based on ECCs and using cAHFs based on TCR hash functions from Section 3.3.1. We do not compare Yamada's third VRF here, because it follows a more generic approach and is not aimed at being particularly efficient. However, we will discuss it in detail in Chapter 4 since this generic approach enables reductions with optimal tightness. Overall, our comparison shows that instantiating the VRFs with cAHFs *significantly reduces the key and proof sizes*. Furthermore, comparing our new VRF in Construction 1 to any of the other VRFs shows that our new VRF has *either significantly smaller secret keys, verification keys or proofs than each other VRF*. Moreover, our VRF based on blockwise partitioning in Construction 2 is by far the most efficient VRF from all VRFs that we compared. However, note that our VRF based on blockwise partitioning in Construction 2 can only be compared to the other VRFs to a limited degree because its security is based on a $q$-$\mathsf{DBDHI}$ assumption with a much larger $q$, which makes the assumption significantly stronger as Cheon has shown [Che06].

Formulas for key and proof sizes. Our comparison is in the concrete setting instead of an asymptotic one. That is, we consider realistic runtime, advantage and security parameters for the comparison. Table 3.2 thus shows the sizes of the verification keys $|\mathsf{vk}|$ and proofs $|\pi|$ as the number of group elements they contain as functions of $\lambda, Q, \varepsilon, \delta$ and $t$. Analogously, $|\mathsf{sk}|$ shows the number of $\mathbb{Z}_p$ elements the secretkey contains in dependence of $\lambda, Q, \varepsilon, \delta$ and $t$. The caption precisely explains how the key and proof sizes relate to these variables. Furthermore, the table shows the advantage of the solver $\mathrm{Adv}_{\mathcal{B}}$ against the respective hard problem. For our VRF in Construction 2, the VRFs of [Kat17] and [Yam17a], this is the

$q$-DBDHI assumption introduced by Boneh and Boyen [BB04a]. For our VRF in Construction 1 and Jager's VRF [Jag15], this is the $q$-DDH assumption. While the security of Construction 2 is based on the $q$-DBDHI assumption with a much larger and is thus only comparable to a limited degree [Che06], the key and proof sizes of all other constructions are comparable for the following reasons, even though they rely on different assumptions.

1. Cheon's algorithm [Che06] is the most efficient known generic algorithm to solve both the $q$-DDH assumption and the $q$-DBDHI assumption.

2. Except for Yamada's VRFs [Yam17a], all schemes considered share almost the same $q$ in the assumption. Also, Yamada's VRF relies on a $\mathcal{O}(\lambda \log(\lambda))$-DBDHI assumption [Yam17a], and is therefore reasonably close the $q$ of the assumptions of the other schemes. Hence, Yamada's VRFs would only need to use slightly larger groups to compensate the stronger assumption.

Instantiation choices. The concrete number of group elements in keys and proofs depends on some instantiation choices, which we describe here. Since the VRF instantiation with cAHFs takes inputs from $\{0,1\}^*$, we level the playing field by assuming that the instantiations using ECCs first hash the inputs with a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^n$ and thus the VRFs also take inputs from $\{0,1\}^*$. We let $n = 2\lambda$, to ensure the collision resistance of $H$ against birthday attacks. Hence, when an ECC $C$ is used in an instantiation, then $C$ is always chosen to take inputs from $\{0,1\}^{2\lambda}$. We list the key sizes of the different VRFs, instantiated with cAHFs and with ECCs in Table 3.3.

Unfortunately, assessing the potential efficiency of the instantiation using bAHFs with ECCs is only possible to a limited degree because finding the best known ECC for a given input length and relative minimal distance is non-trivial for larger numbers. Code tables [Gra07], the most extensive collection (to the best of our knowledge) of best known codes for different parameters, only lists binary codes of length up to 256, which is too small for reasonable security parameters. Therefore, we compare the instantiation of bAHFs with *Bose–Chaudhuri–Hocquenghem* (BCH) codes and with hypothetical ECCs on the MRRW and GV bound. We describe these potential instanations more specifically below.

- We consider primitive *Bose–Chaudhuri–Hocquenghem* (BCH) codes that we puncture to achieve the desired relative minimal distance. Again, tables in for example [PW88, Table 9.1] only list codes for lengths up to 1023. Therefore, we wrote a small program that finds the most suited primitive BCH code for this purpose. It can be found in Appendix A or at `https://github.com/DavidNiehues/bch-code-search`. The program considers the Bose distance of the BCH codes instead of the (sometimes worse) design distance. The caption of Table 3.3 states the used primitive BCH code explicitly.

- Furthermore, we present key and proof sizes under the assumptions that ECCs on the GV and MRRW bound can be efficiently instantiated.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert$ $\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| Sec. 5.1 in [Kat17] | ECCs | $3+\zeta\eta^{\mathsf{bAHF}}$ | $\eta^{\mathsf{bAHF}}\zeta+1$ | $\eta^{\mathsf{bAHF}}+\eta^{\mathsf{bAHF}}n^{\mathsf{ECC}}+\zeta+1$ | $\tau^{\mathsf{bAHF}}+\mathsf{negl}$ |
| | cAHF | $3+\zeta^{\mathsf{hash}}\eta^{\mathsf{hash}}$ | $\zeta^{\mathsf{hash}}+1$ | $\eta^{\mathsf{hash}}+\eta^{\mathsf{hash}}(n^{\mathsf{hash}})+\zeta^{\mathsf{hash}}+1$ | $\tau^{\mathsf{tcrh}}+\mathsf{negl}$ |
| Sec. 5.3 in [Kat17] | ECCs | $3+\eta^{\mathsf{bAHF}}(2^{\zeta/2+2}-2)$ | $\eta^{\mathsf{bAHF}}\zeta+1$ | $2\eta^{\mathsf{bAHF}}-1$ | $\tau^{\mathsf{bAHF}}+\mathsf{negl}$ |
| | cAHF | $3+\eta^{\mathsf{hash}}(2^{\zeta^{\mathsf{hash}}/2+2}-2)$ | $\eta^{\mathsf{hash}}\zeta^{\mathsf{hash}}+1$ | $2\eta^{\mathsf{hash}}-1$ | $\tau^{\mathsf{tcrh}}+\mathsf{negl}$ |
| Sec. 6.1 in [Yam17a] | ECCs | $\eta^{\mathsf{bAHF}}n_1^{\mathsf{ECC}}+2$ | $\eta^{\mathsf{bAHF}}$ | $\eta^{\mathsf{bAHF}}n_2^{\mathsf{ECC}}$ | $\tau^{\mathsf{bAHF}}+\mathsf{negl}$ |
| | cAHF | $\eta^{\mathsf{hash}}n_1^{\mathsf{hash}}+2$ | $\eta^{\mathsf{hash}}$ | $\eta^{\mathsf{hash}}n_2^{\mathsf{hash}}$ | $\tau^{\mathsf{tcrh}}+\mathsf{negl}$ |
| Sec. 6.3 in [Yam17a] | ECCs | $\eta^{\mathsf{bAHF}}+2$ | $\eta^{\mathsf{bAHF}}$ | $\eta^{\mathsf{bAHF}}(n_1^{\mathsf{ECC}}+n_2^{\mathsf{ECC}}-1)$ | $\tau^{\mathsf{bAHF}}+\mathsf{negl}$ |
| | cAHF | $\eta^{\mathsf{hash}}+2$ | $\eta^{\mathsf{hash}}$ | $\eta^{\mathsf{hash}}(n_1^{\mathsf{hash}}+n_2^{\mathsf{hash}}-1)$ | $\tau^{\mathsf{tcrh}}+\mathsf{negl}$ |
| [Jag15] | ECCs | $2n^{\mathsf{ECC}}+2$ | $2n^{\mathsf{ECC}}$ | $n^{\mathsf{ECC}}$ | $\tau^{\mathsf{bAHF}}$ |
| | cAHFs | $2n^{\mathsf{hash}}+2$ | $2n^{\mathsf{hash}}$ | $n^{\mathsf{hash}}$ | $\tau^{\mathsf{tcrh}}$ |
| Construction 1 | ECCs | $n^{\mathsf{ECC}}+4$ | $n^{\mathsf{ECC}}+2$ | $n^{\mathsf{ECC}}+1$ | $\tau^{\mathsf{bAHF}}$ |
| | cAHFs (as shown) | $n^{\mathsf{hash}}+4$ | $n^{\mathsf{hash}}+2$ | $n^{\mathsf{hash}}+1$ | $\tau^{\mathsf{tcrh}}$ |
| Construction 2 | blockwise partitioning | $\lfloor\log n^{\mathsf{hash}}\rfloor+3$ | $\lfloor\log n^{\mathsf{hash}}\rfloor+1$ | $\lfloor\log n^{\mathsf{hash}}\rfloor+1$ | $\tau^{\mathsf{blk}}+\mathsf{negl}$ |

*Table 3.2:* The sizes of $\mathsf{vk}$, $\mathsf{sk}$, $\pi$ as the number of elements from $\mathbb{G}$ and $\mathbb{Z}_p$, respectively, and the advantage of the solver in the security proof for the instantiation of our VRF and the VRFs of Jager [Jag15], Yamada [Yam17a] and Katsumata [Kat17]. We denote with $n^{\mathsf{type}} \in \mathbb{N}$, for $\mathsf{type} \in \{\mathsf{ecc}, \mathsf{hash}\}$ the number of bits in the output or the hash function used in the construction. Moreover, $\eta^{\mathsf{bAHF}} := \left\lceil \log_{1-\delta}\left(\frac{-\varepsilon\cdot\ln(2)}{(\varepsilon+1/2)\cdot Q\cdot\ln((1-\delta)/2)}\right)\right\rceil$ is the number of positions in the key of the bAHF that are not $\perp$. Note that this is the optimal choice for $\eta^{\mathsf{bAHF}}$ that we discussed in Lemma 3. Analogously, $\eta^{\mathsf{hash}} := \lceil\log(4t(2t-1)/\varepsilon)\rceil$ is the number of positions that are not $\perp$ in $\mathsf{K}$ both for cAHFs (see Theorem 4) as well as for blockwise partitioning (see Lemma 6). For the VRFs from Katsumata [Kat17], $\zeta^{\mathsf{type}} = \lfloor\log(2n^{\mathsf{type}})\rfloor + 1$ is the number of bits required to encode an element from $[n^{\mathsf{type}}]$, again for $\mathsf{type} \in \{\mathsf{ecc}, \mathsf{hash}\}$. $\tau^{\mathsf{bAHF}}$ is as in Definition 16 and describes the advantage of a solver against the underlying $q$-type assumption. $\mathsf{negl}$ represents statistically negligible values introduced in the security proofs. Note that for Yamada's VRFs [Yam17a], $n_1^{\mathsf{ECC}}$, $n_2^{\mathsf{ECC}} \in \mathbb{N}$ can be chosen freely such that $n^{\mathsf{ECC}} = n_1^{\mathsf{ECC}} n_2^{\mathsf{ECC}}$. Finally, we have $\tau^{\mathsf{tcrh}} = \tau^{\mathsf{blk}} = \varepsilon^2/(32t^2 - 16t)$ from Theorem 4 and Lemma 6, respectively.

Assuming instantiations with ECCs on the GV and MRRW bound gives the instantiation with bAHFs and ECCs an advantage over instantiations using cAHFs with TCRs, since ECCs on the MRRW bound are the best theoretically possible ECCs and it is not known whether ECCs on the GV bound can be constructed efficiently. We consider both bounds because there are concrete codes better than the GV bound. Hence, only showing codes on the GV bound would be unfair to the instantiation with ECCs.

In the calculation of the key sizes of Yamada's VRFs [Yam17a], we pick $n_1^{\mathsf{ECC}} = n_2^{\mathsf{ECC}}$ as $\left\lceil \sqrt{n^{\mathsf{ECC}}} \right\rceil$ in order to make the parameter sizes comparable.

Kohl's VRF with short proofs. In addition to the VRFs that we discuss in detail, we also want to discuss Kohl's VRF [Koh19] with short proofs of $\omega(1)$ many group elements from a standard assumption. In contrast to the other VRFs that we compare, Kohl's VRF relies on an AHF together with the artificial abort technique, which was first introduced by Waters [Wat05]. Furthermore, it uses ECCs over larger finite fields of polynomial size, as first discussed for partitioning by Bitansky in the full version [Bit17b] of [Bit17a]. The full version was later published as [Bit20]. While using an ECC over a larger build can be used in a bAHF, as Bitansky has shown [Bit20, Section 4.1.1], Kohl's VRF relies on a particular choice of $\eta$ to play out its strength of achieving short proof sizes at the cost of a larger loss, which we use to normalize the comparison in Table 3.3. Therefore, we do not include it in our detailed comparison of VRFs in Table 3.3. Nontheless, we want to give an impression of the size of the proofs of Kohl's VRF if it is instantiated exactly as described by Kohl in [Koh19].

We are considering the same scenario as in Table 3.3. The parameters for the instantiation of Kohl's VRF are $\lambda = 128$ and $Q = 2^{25}$. Then the proofs of Kohl's VRF have a size of $|\pi| = 3 \cdot (\lceil \log(2Q)/(\nu \log(\lambda)) \rceil + 1)$ many group elements, where $0 < \nu \leq 1$ is a parameter that allows a tradeoff between the size of the proofs and the size of the verification keys of the VRF [Koh19, Section 3]. Note that this size of proofs is asymptotically $\omega(1)$ because $Q$ is polynomial in $\lambda$. We show in Table 3.4 how many group elements the proofs of Kohl's VRF have in this setting for different values of $\nu$.

Smaller keys and proofs using cAHFs. Table 3.3 shows the concrete number of group elements of the different instantiations in the setting with $\lambda = 128$, $Q = 2^{25}$, $t = 2^{50}$ and $\varepsilon = 2^{-25}$. The instantiation of the VRFs with cAHFs improves the size of keys and proofs significantly, even compared to bAHFs *instantiated with the best theoretically possible ECCs* on the MRRW bound. Concretely, using cAHFs with TCRs instead of bAHFs with ECCs on the MRRW bound *reduces the size of the proofs of the VRF in Section 5.1 in [Kat17] by* $\approx 61\%$ in the setting of Table 3.3. Compared to an instantiation with ECCs on the GV bound, we reduce the proof size by $\approx 78\%$. Compared to the instantitation with punctured primitive BCH codes, the improvement is $\approx 87\%$. Particularly, keys and proofs whose size

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 1551 | 1549 | 261883 | $\approx 2^{-155}$ |
| | GV | 1551 | 1549 | 154942 | $\approx 2^{-155}$ |
| | MRRW | 1422 | 1420 | 85797 | $\approx 2^{-155}$ |
| | cAHF | 1283 | 1281 | 33291 | $\approx 2^{-155}$ |
| [Kat17] Sec. 5.3 | BCH | 32769 | 1549 | 257 | $\approx 2^{-155}$ |
| | GV | 32769 | 1549 | 257 | $\approx 2^{-155}$ |
| | MRRW | 23094 | 1420 | 257 | $\approx 2^{-155}$ |
| | cAHF | 16131 | 1281 | 255 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.1 | BCH | 5936 | 129 | 5934 | $\approx 2^{-155}$ |
| | GV | 4517 | 129 | 4515 | $\approx 2^{-155}$ |
| | MRRW | 3356 | 129 | 3354 | $\approx 2^{-155}$ |
| | cAHF | 2178 | 128 | 2176 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.2 | BCH | 131 | 129 | 11739 | $\approx 2^{-155}$ |
| | GV | 131 | 129 | 8901 | $\approx 2^{-155}$ |
| | MRRW | 131 | 129 | 6579 | $\approx 2^{-155}$ |
| | cAHF | 130 | 128 | 4224 | $\approx 2^{-155}$ |
| [Jag15] | BCH | 4060 | 4058 | 2029 | $\approx 2^{-155}$ |
| | GV | 2402 | 2400 | 1200 | $\approx 2^{-155}$ |
| | MRRW | 1330 | 1328 | 664 | $\approx 2^{-155}$ |
| | cAHF | 520 | 518 | 259 | $\approx 2^{-155}$ |
| Construction 1 | BCH | 2033 | 2031 | 2030 | $\approx 2^{-155}$ |
| | GV | 1204 | 1202 | 1201 | $\approx 2^{-155}$ |
| | MRRW | 668 | 666 | 665 | $\approx 2^{-155}$ |
| | cAHF | 263 | 261 | 260 | $\approx 2^{-155}$ |
| Construction 2 | Blockwise Part. | 11 | 9 | 9 | $\approx 2^{-155}$ |

*Table 3.3:* Key and proof sizes for $\lambda = 128, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-25}$ and $\delta = 0.235$. In consequence, we have $\eta^{\mathsf{bAHF}} = 129$. Puncturing a primitive $[2047, 264, 495]$ BCH-code 18 times to a $[2029, 264, 477]_2$ code yields $n^{\mathsf{BCH}} = 2029, n_1^{\mathsf{BCH}} = 46, n_2^{\mathsf{BCH}} = 46$ and $\zeta^{\mathsf{BCH}} = 12$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 1200, n_1^{\mathsf{GV}} = 35, n_2^{\mathsf{GV}} = 35$ and $\zeta^{\mathsf{GV}} = 12$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 664, n_1^{\mathsf{MRRW}} = 26, n_2^{\mathsf{MRRW}} = 26$ and $\zeta^{\mathsf{MRRW}} = 11$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 259, n_1^{\mathsf{hash}} = 17, n_2^{\mathsf{hash}} = 17, \eta^{\mathsf{hash}} = 128$ and $\zeta^{\mathsf{hash}} = 10$.

| $\nu$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert\pi\rvert$ | 117 | 60 | 42 | 33 | 27 | 24 | 21 | 18 | 18 | 15 |

*Table 3.4:* The size of proofs of Kohl's VRF for $\lambda = 128$, $Q = 2^{25}$ and for $\nu \in \{0.1, 0.2, \ldots, 0.9, 1\}$.

depends linearly on $n^{\mathsf{ECC}}$ shrink when the VRFs are instantiated with cAHFs. Over all key and proof sizes affected by the improvement, the reduction amounts for at least 9% of the size of an instantiation with ECCs on the MRRW bound. Note that the size of all keys and proofs stays at least the same. Hence, by making an additional (but from a practical point of view plausible and natural) complexity assumption, we can reduce the key and proof sizes significantly, which may be useful for many practical applications of VRFs. Furthermore, it hints at the usefulness of cAHFs for other primitives.

Efficiency of $\mathcal{VRF}_{\mathsf{cAHF}}$. Table 3.3 shows that Construction 1 nearly halves the size of the verification and secret key of Jager's VRF while maintaining the proof size. Comparing Construction 1 with the VRFs of Katsumata and Yamada shows that, in the worst case, Construction 1 has secret or verification keys about twice the size of the verification or secret keys of VRFs by Katsumata and Yamada. At the same time, when instantiated with cAHFs, our VRF always has either verification or secret keys with size only $\approx 20\%$ of the size of the respective keys of the VRFs of Yamada and Katsumata with the same instantiation.

Efficiency of $\mathcal{VRF}_{\mathsf{Blk}}$. The comparison in Table 3.3 shows that our VRF based on blockwise partitioning in Construction 2 is by far the most efficient VRF from those that we considered. Even though we have to take into account that Construction 2 is based on a significantly stronger assumption than the other VRFs that we considered, the VRF would still be the most efficient construction after compensating the stronger assumption by using larger groups.

Further comparisons. Below, we provide more comparisons like those in Table 3.3. These comparisons further support the efficiency of cAHFs, blockwise partitioning and both our VRFs. The results we present are calculated using the formulas stated in Table 3.2. Compared to Table 3.3, we provide key and proof sizes for $\lambda \in \{100, 128, 256\}$ and $\varepsilon \in \{2^{-25}, 2^{-50}\}$. For every combination of $\lambda$ and $\varepsilon$, $\delta$ is chosen such that the advantages for the instantiation using ECCs and the instantiation using truncation-collision resistant hash functions are approximately the same. The results show that using the cAHF instantiated with a truncation-collision resistant hash function reduces key and proof sizes significantly. Furthermore, our VRF from blockwise partitioning is by far the most efficient in all settings, however, also requires the strongest assumptions.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 2005 | 2003 | 567966 | $\approx 2^{-205}$ |
| | GV | 1851 | 1849 | 225931 | $\approx 2^{-205}$ |
| | MRRW | 1697 | 1695 | 110892 | $\approx 2^{-205}$ |
| | cAHF | 1380 | 1378 | 31222 | $\approx 2^{-205}$ |
| [Kat17] Sec. 5.3 | BCH | 55443 | 2003 | 307 | $\approx 2^{-205}$ |
| | GV | 39119 | 1849 | 307 | $\approx 2^{-205}$ |
| | MRRW | 27569 | 1695 | 307 | $\approx 2^{-205}$ |
| | cAHF | 13467 | 1378 | 305 | $\approx 2^{-205}$ |
| [Yam17a] Sec. 6.1 | BCH | 9396 | 154 | 9394 | $\approx 2^{-205}$ |
| | GV | 6008 | 154 | 6006 | $\approx 2^{-205}$ |
| | MRRW | 4160 | 154 | 4158 | $\approx 2^{-205}$ |
| | cAHF | 2297 | 153 | 2295 | $\approx 2^{-205}$ |
| [Yam17a] Sec. 6.2 | BCH | 156 | 154 | 18634 | $\approx 2^{-205}$ |
| | GV | 156 | 154 | 11858 | $\approx 2^{-205}$ |
| | MRRW | 156 | 154 | 8162 | $\approx 2^{-205}$ |
| | cAHF | 155 | 153 | 4437 | $\approx 2^{-205}$ |
| [Jag15] | BCH | 7376 | 7374 | 3687 | $\approx 2^{-205}$ |
| | GV | 2934 | 2932 | 1466 | $\approx 2^{-205}$ |
| | MRRW | 1440 | 1438 | 719 | $\approx 2^{-205}$ |
| | cAHF | 408 | 406 | 203 | $\approx 2^{-205}$ |
| Construction 1 | BCH | 3691 | 3689 | 3688 | $\approx 2^{-205}$ |
| | GV | 1470 | 1468 | 1467 | $\approx 2^{-205}$ |
| | MRRW | 723 | 721 | 720 | $\approx 2^{-205}$ |
| | cAHF | 207 | 205 | 204 | $\approx 2^{-205}$ |
| Construction 2 | Blockwise Part. | 10 | 8 | 8 | $\approx 2^{-205}$ |

*Table 3.5:* Key and proof sizes for $\lambda = 100, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-50}$ and $\delta = 0.286$. In consequence, we have $\eta^{\mathsf{bAHF}} = 154$. Puncturing a primitive $[4095, 211, 1463]$ BCH-code 408 times to a $[3687, 211, 1055]_2$ code yields $n^{\mathsf{BCH}} = 3687, n_1^{\mathsf{BCH}} = 61, n_2^{\mathsf{BCH}} = 61$ and $\zeta^{\mathsf{BCH}} = 13$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 1466, n_1^{\mathsf{GV}} = 39, n_2^{\mathsf{GV}} = 39$ and $\zeta^{\mathsf{GV}} = 12$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 719, n_1^{\mathsf{MRRW}} = 27, n_2^{\mathsf{MRRW}} = 27$ and $\zeta^{\mathsf{MRRW}} = 11$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 203, n_1^{\mathsf{hash}} = 15, n_2^{\mathsf{hash}} = 15, \eta^{\mathsf{hash}} = 153$ and $\zeta^{\mathsf{hash}} = 9$.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 1551 | 1549 | 259174 | $\approx 2^{-155}$ |
| | GV | 1422 | 1420 | 121143 | $\approx 2^{-155}$ |
| | MRRW | 1422 | 1420 | 67092 | $\approx 2^{-155}$ |
| | cAHF | 1155 | 1153 | 26122 | $\approx 2^{-155}$ |
| [Kat17] Sec. 5.3 | BCH | 32769 | 1549 | 257 | $\approx 2^{-155}$ |
| | GV | 23094 | 1420 | 257 | $\approx 2^{-155}$ |
| | MRRW | 23094 | 1420 | 257 | $\approx 2^{-155}$ |
| | cAHF | 11267 | 1153 | 255 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.1 | BCH | 5807 | 129 | 5805 | $\approx 2^{-155}$ |
| | GV | 4001 | 129 | 3999 | $\approx 2^{-155}$ |
| | MRRW | 2969 | 129 | 2967 | $\approx 2^{-155}$ |
| | cAHF | 1922 | 128 | 1920 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.2 | BCH | 131 | 129 | 11481 | $\approx 2^{-155}$ |
| | GV | 131 | 129 | 7869 | $\approx 2^{-155}$ |
| | MRRW | 131 | 129 | 5805 | $\approx 2^{-155}$ |
| | cAHF | 130 | 128 | 3712 | $\approx 2^{-155}$ |
| [Jag15] | BCH | 4018 | 4016 | 2008 | $\approx 2^{-155}$ |
| | GV | 1878 | 1876 | 938 | $\approx 2^{-155}$ |
| | MRRW | 1040 | 1038 | 519 | $\approx 2^{-155}$ |
| | cAHF | 408 | 406 | 203 | $\approx 2^{-155}$ |
| Construction 1 | BCH | 2012 | 2010 | 2009 | $\approx 2^{-155}$ |
| | GV | 942 | 940 | 939 | $\approx 2^{-155}$ |
| | MRRW | 523 | 521 | 520 | $\approx 2^{-155}$ |
| | cAHF | 207 | 205 | 204 | $\approx 2^{-155}$ |
| Construction 2 | Blockwise Part. | 10 | 8 | 8 | $\approx 2^{-155}$ |

*Table 3.6:* Key and proof sizes for $\lambda = 100, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-25}$ and $\delta = 0.235$. In consequence, we have $\eta^{\mathsf{bAHF}} = 129$. Puncturing a primitive $[2047, 209, 511]$ BCH-code 39 times to a $[2008, 209, 472]_2$ code yields $n^{\mathsf{BCH}} = 2008, n_1^{\mathsf{BCH}} = 45, n_2^{\mathsf{BCH}} = 45$ and $\zeta^{\mathsf{BCH}} = 12$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 938, n_1^{\mathsf{GV}} = 31, n_2^{\mathsf{GV}} = 31$ and $\zeta^{\mathsf{GV}} = 11$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 519, n_1^{\mathsf{MRRW}} = 23, n_2^{\mathsf{MRRW}} = 23$ and $\zeta^{\mathsf{MRRW}} = 11$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 203, n_1^{\mathsf{hash}} = 15, n_2^{\mathsf{hash}} = 15, \eta^{\mathsf{hash}} = 128$ and $\zeta^{\mathsf{hash}} = 9$.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 2005 | 2003 | 581672 | $\approx 2^{-205}$ |
| | GV | 1851 | 1849 | 289071 | $\approx 2^{-205}$ |
| | MRRW | 1697 | 1695 | 141846 | $\approx 2^{-205}$ |
| | cAHF | 1533 | 1531 | 39791 | $\approx 2^{-205}$ |
| [Kat17] Sec. 5.3 | BCH | 55443 | 2003 | 307 | $\approx 2^{-205}$ |
| | GV | 39119 | 1849 | 307 | $\approx 2^{-205}$ |
| | MRRW | 27569 | 1695 | 307 | $\approx 2^{-205}$ |
| | cAHF | 19281 | 1531 | 305 | $\approx 2^{-205}$ |
| [Yam17a] Sec. 6.1 | BCH | 9550 | 154 | 9548 | $\approx 2^{-205}$ |
| | GV | 6778 | 154 | 6776 | $\approx 2^{-205}$ |
| | MRRW | 4776 | 154 | 4774 | $\approx 2^{-205}$ |
| | cAHF | 2603 | 153 | 2601 | $\approx 2^{-205}$ |
| [Yam17a] Sec. 6.2 | BCH | 156 | 154 | 18942 | $\approx 2^{-205}$ |
| | GV | 156 | 154 | 13398 | $\approx 2^{-205}$ |
| | MRRW | 156 | 154 | 9394 | $\approx 2^{-205}$ |
| | cAHF | 155 | 153 | 5049 | $\approx 2^{-205}$ |
| [Jag15] | BCH | 7554 | 7552 | 3776 | $\approx 2^{-205}$ |
| | GV | 3754 | 3752 | 1876 | $\approx 2^{-205}$ |
| | MRRW | 1842 | 1840 | 920 | $\approx 2^{-205}$ |
| | cAHF | 520 | 518 | 259 | $\approx 2^{-205}$ |
| Construction 1 | BCH | 3780 | 3778 | 3777 | $\approx 2^{-205}$ |
| | GV | 1880 | 1878 | 1877 | $\approx 2^{-205}$ |
| | MRRW | 924 | 922 | 921 | $\approx 2^{-205}$ |
| | cAHF | 263 | 261 | 260 | $\approx 2^{-205}$ |
| Construction 2 | Blockwise Part. | 11 | 9 | 9 | $\approx 2^{-205}$ |

*Table 3.7:* Key and proof sizes for $\lambda = 128, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-50}$ and $\delta = 0.286$. In consequence, we have $\eta^{\mathsf{bAHF}} = 154$. Puncturing a primitive $[4095, 259, 1399]$ BCH-code 319 times to a $[3776, 259, 1080]_2$ code yields $n^{\mathsf{BCH}} = 3776, n_1^{\mathsf{BCH}} = 62, n_2^{\mathsf{BCH}} = 62$ and $\zeta^{\mathsf{BCH}} = 13$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 1876, n_1^{\mathsf{GV}} = 44, n_2^{\mathsf{GV}} = 44$ and $\zeta^{\mathsf{GV}} = 12$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 920, n_1^{\mathsf{MRRW}} = 31, n_2^{\mathsf{MRRW}} = 31$ and $\zeta^{\mathsf{MRRW}} = 11$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 259, n_1^{\mathsf{hash}} = 17, n_2^{\mathsf{hash}} = 17, \eta^{\mathsf{hash}} = 153$ and $\zeta^{\mathsf{hash}} = 10$.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 1551 | 1549 | 261883 | $\approx 2^{-155}$ |
| | GV | 1551 | 1549 | 154942 | $\approx 2^{-155}$ |
| | MRRW | 1422 | 1420 | 85797 | $\approx 2^{-155}$ |
| | cAHF | 1283 | 1281 | 33291 | $\approx 2^{-155}$ |
| [Kat17] Sec. 5.3 | BCH | 32769 | 1549 | 257 | $\approx 2^{-155}$ |
| | GV | 32769 | 1549 | 257 | $\approx 2^{-155}$ |
| | MRRW | 23094 | 1420 | 257 | $\approx 2^{-155}$ |
| | cAHF | 16131 | 1281 | 255 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.1 | BCH | 5936 | 129 | 5934 | $\approx 2^{-155}$ |
| | GV | 4517 | 129 | 4515 | $\approx 2^{-155}$ |
| | MRRW | 3356 | 129 | 3354 | $\approx 2^{-155}$ |
| | cAHF | 2178 | 128 | 2176 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.2 | BCH | 131 | 129 | 11739 | $\approx 2^{-155}$ |
| | GV | 131 | 129 | 8901 | $\approx 2^{-155}$ |
| | MRRW | 131 | 129 | 6579 | $\approx 2^{-155}$ |
| | cAHF | 130 | 128 | 4224 | $\approx 2^{-155}$ |
| [Jag15] | BCH | 4060 | 4058 | 2029 | $\approx 2^{-155}$ |
| | GV | 2402 | 2400 | 1200 | $\approx 2^{-155}$ |
| | MRRW | 1330 | 1328 | 664 | $\approx 2^{-155}$ |
| | cAHF | 520 | 518 | 259 | $\approx 2^{-155}$ |
| Construction 1 | BCH | 2033 | 2031 | 2030 | $\approx 2^{-155}$ |
| | GV | 1204 | 1202 | 1201 | $\approx 2^{-155}$ |
| | MRRW | 668 | 666 | 665 | $\approx 2^{-155}$ |
| | cAHF | 263 | 261 | 260 | $\approx 2^{-155}$ |
| Construction 2 | Blockwise Part. | 11 | 9 | 9 | $\approx 2^{-155}$ |

*Table 3.8:* Key and proof sizes for $\lambda = 128, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-25}$ and $\delta = 0.235$. In consequence, we have $\eta^{\mathsf{bAHF}} = 129$. Puncturing a primitive $[2047, 264, 495]$ BCH-code 18 times to a $[2029, 264, 477]_2$ code yields $n^{\mathsf{BCH}} = 2029, n_1^{\mathsf{BCH}} = 46, n_2^{\mathsf{BCH}} = 46$ and $\zeta^{\mathsf{BCH}} = 12$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 1200, n_1^{\mathsf{GV}} = 35, n_2^{\mathsf{GV}} = 35$ and $\zeta^{\mathsf{GV}} = 12$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 664, n_1^{\mathsf{MRRW}} = 26, n_2^{\mathsf{MRRW}} = 26$ and $\zeta^{\mathsf{MRRW}} = 11$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 259, n_1^{\mathsf{hash}} = 17, n_2^{\mathsf{hash}} = 17, \eta^{\mathsf{hash}} = 128$ and $\zeta^{\mathsf{hash}} = 10$.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 2159 | 2157 | 1177961 | $\approx 2^{-205}$ |
| | GV | 2005 | 2003 | 577822 | $\approx 2^{-205}$ |
| | MRRW | 1851 | 1849 | 283527 | $\approx 2^{-205}$ |
| | cAHF | 1686 | 1684 | 78960 | $\approx 2^{-205}$ |
| [Kat17] Sec. 5.3 | BCH | 78543 | 2157 | 307 | $\approx 2^{-205}$ |
| | GV | 55443 | 2003 | 307 | $\approx 2^{-205}$ |
| | MRRW | 39119 | 1849 | 307 | $\approx 2^{-205}$ |
| | cAHF | 27390 | 1684 | 305 | $\approx 2^{-205}$ |
| [Yam17a] Sec. 6.1 | BCH | 13554 | 154 | 13552 | $\approx 2^{-205}$ |
| | GV | 9550 | 154 | 9548 | $\approx 2^{-205}$ |
| | MRRW | 6624 | 154 | 6622 | $\approx 2^{-205}$ |
| | cAHF | 3521 | 153 | 3519 | $\approx 2^{-205}$ |
| [Yam17a] Sec. 6.2 | BCH | 156 | 154 | 26950 | $\approx 2^{-205}$ |
| | GV | 156 | 154 | 18942 | $\approx 2^{-205}$ |
| | MRRW | 156 | 154 | 13090 | $\approx 2^{-205}$ |
| | cAHF | 155 | 153 | 6885 | $\approx 2^{-205}$ |
| [Jag15] | BCH | 15298 | 15296 | 7648 | $\approx 2^{-205}$ |
| | GV | 7504 | 7502 | 3751 | $\approx 2^{-205}$ |
| | MRRW | 3682 | 3680 | 1840 | $\approx 2^{-205}$ |
| | cAHF | 1032 | 1030 | 515 | $\approx 2^{-205}$ |
| Construction 1 | BCH | 7652 | 7650 | 7649 | $\approx 2^{-205}$ |
| | GV | 3755 | 3753 | 3752 | $\approx 2^{-205}$ |
| | MRRW | 1844 | 1842 | 1841 | $\approx 2^{-205}$ |
| | cAHF | 519 | 517 | 516 | $\approx 2^{-205}$ |
| Construction 2 | Blockwise Part. | 12 | 10 | 10 | $\approx 2^{-205}$ |

*Table 3.9:* Key and proof sizes for $\lambda = 256, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-50}$ and $\delta = 0.286$. In consequence, we have $\eta^{\mathsf{bAHF}} = 154$. Puncturing a primitive $[8191, 521, 2731]$ BCH-code 543 times to a $[7648, 521, 2188]_2$ code yields $n^{\mathsf{BCH}} = 7648, n_1^{\mathsf{BCH}} = 88, n_2^{\mathsf{BCH}} = 88$ and $\zeta^{\mathsf{BCH}} = 14$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 3751, n_1^{\mathsf{GV}} = 62, n_2^{\mathsf{GV}} = 62$ and $\zeta^{\mathsf{GV}} = 13$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 1840, n_1^{\mathsf{MRRW}} = 43, n_2^{\mathsf{MRRW}} = 43$ and $\zeta^{\mathsf{MRRW}} = 12$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 515, n_1^{\mathsf{hash}} = 23, n_2^{\mathsf{hash}} = 23, \eta^{\mathsf{hash}} = 153$ and $\zeta^{\mathsf{hash}} = 11$.

| Construction | Instantiation | $\lvert\mathsf{vk}\rvert$ $\#\mathbb{G}$ | $\lvert\mathsf{sk}\rvert$ $\#\mathbb{Z}_p$ | $\lvert\pi\rvert\#\mathbb{G}$ | $\mathrm{Adv}_{\mathcal{B}}$ |
|---|---|---|---|---|---|
| [Kat17] Sec. 5.1 | BCH | 1809 | 1807 | 920946 | $\approx 2^{-155}$ |
| | GV | 1680 | 1678 | 309743 | $\approx 2^{-155}$ |
| | MRRW | 1551 | 1549 | 171454 | $\approx 2^{-155}$ |
| | cAHF | 1411 | 1409 | 66060 | $\approx 2^{-155}$ |
| [Kat17] Sec. 5.3 | BCH | 65793 | 1807 | 257 | $\approx 2^{-155}$ |
| | GV | 46443 | 1678 | 257 | $\approx 2^{-155}$ |
| | MRRW | 32769 | 1549 | 257 | $\approx 2^{-155}$ |
| | cAHF | 22915 | 1409 | 255 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.1 | BCH | 10967 | 129 | 10965 | $\approx 2^{-155}$ |
| | GV | 6323 | 129 | 6321 | $\approx 2^{-155}$ |
| | MRRW | 4775 | 129 | 4773 | $\approx 2^{-155}$ |
| | cAHF | 2946 | 128 | 2944 | $\approx 2^{-155}$ |
| [Yam17a] Sec. 6.2 | BCH | 131 | 129 | 21801 | $\approx 2^{-155}$ |
| | GV | 131 | 129 | 12513 | $\approx 2^{-155}$ |
| | MRRW | 131 | 129 | 9417 | $\approx 2^{-155}$ |
| | cAHF | 130 | 128 | 5760 | $\approx 2^{-155}$ |
| [Jag15] | BCH | 14278 | 14276 | 7138 | $\approx 2^{-155}$ |
| | GV | 4802 | 4800 | 2400 | $\approx 2^{-155}$ |
| | MRRW | 2658 | 2656 | 1328 | $\approx 2^{-155}$ |
| | cAHF | 1032 | 1030 | 515 | $\approx 2^{-155}$ |
| Construction 1 | BCH | 7142 | 7140 | 7139 | $\approx 2^{-155}$ |
| | GV | 2404 | 2402 | 2401 | $\approx 2^{-155}$ |
| | MRRW | 1332 | 1330 | 1329 | $\approx 2^{-155}$ |
| | cAHF | 519 | 517 | 516 | $\approx 2^{-155}$ |
| Construction 2 | Blockwise Part. | 12 | 10 | 10 | $\approx 2^{-155}$ |

*Table 3.10:* Key and proof sizes for $\lambda = 256, Q = 2^{25}, t = 2^{50}, \varepsilon = 2^{-25}$ and $\delta = 0.235$. In consequence, we have $\eta^{\mathsf{bAHF}} = 129$. Puncturing a primitive $[8191, 520, 2731]$ BCH-code 1053 times to a $[7138, 520, 1678]_2$ code yields $n^{\mathsf{BCH}} = 7138, n_1^{\mathsf{BCH}} = 85, n_2^{\mathsf{BCH}} = 85$ and $\zeta^{\mathsf{BCH}} = 14$. If an ECC on the GV bound is used, this implies $n^{\mathsf{GV}} = 2400, n_1^{\mathsf{GV}} = 49, n_2^{\mathsf{GV}} = 49$ and $\zeta^{\mathsf{GV}} = 13$. Analogously, if an ECC on the MRRW bound is used, this implies $n^{\mathsf{MRRW}} = 1328, n_1^{\mathsf{MRRW}} = 37, n_2^{\mathsf{MRRW}} = 37$ and $\zeta^{\mathsf{MRRW}} = 12$. Finally, if the VRFs are instantiated with a cAHF using TCRs, we have $n^{\mathsf{hash}} = 515, n_1^{\mathsf{hash}} = 23, n_2^{\mathsf{hash}} = 23, \eta^{\mathsf{hash}} = 128$ and $\zeta^{\mathsf{hash}} = 11$.

## 3.6 Conclusion and Discussion

In this chapter, we first discussed the inherent limitations of (balanced) AHFs due to their instantiation from ECCs in Section 3.2. In Section 3.3, we then introduced cAHFs from truncation-collision resistant hash functions as a more efficient drop-in replacement for (balanced) AHFs and showed how it can be used in Jager's VRF [Jag15]. Moreover, note that $\mathcal{VRF}_{\mathsf{cAHF}}$ that we present in Construction 1 is a more efficient variant of Jager's VRF and is only contained in the full version [JN19b] and not in the published version [JN19a]. Next, in Section 3.4, we introduced blockwise partitioning from weak near-collision resistant hash functions as a new semi-generic technique to construct as efficient VRFs as possible without relying on the ROM. Finally, in Section 3.5 we compared the VRFs we presented with the VRFs from previous publications in detail. From this comparison, we conclude that cAHFs and blockwise partitioning offer significant advantages over bAHFs in constructing VRFs. Furthermore, we note that blockwise partitioning is also applicable when constructing IB-KEMs [JKN21]. The two constructions of IB-KEMs from pairings were contributed by my two co-authors and are thus not presented in this thesis. However, we will present the construction of an IB-KEM based on lattices from blockwise partitioning in Chapter 5.

### 3.6.1 Open Questions

Even though we made significant progress in constructing efficient VRFs from pairings, there still remain open research questions that we want to discuss here.

**Research Question 1.** *Are there VRFs that are secure (under standard assumptions) in the standard model and are as efficient as VRFs in the ROM?*

While there are highly efficient VRFs that are about to be standardized [GRPV21], all of these constructions are only secure in the ROM. And even though, as Table 3.3 shows, $\mathcal{VRF}_{\mathsf{Blk}}$ comes close to practical efficiency, it is still significantly less efficient in several aspects than the VRFs specified in [GRPV21]. That is, it has still significantly larger verification/secret keys and proofs. But furthermore, both the evaluation as well as the verification require the time-consuming computation of the bilinear pairing $e$. Thus, even though we made a plausible but non-standard hash function assumption with weak near-collision resistant, our most efficient VRF in $\mathcal{VRF}_{\mathsf{Blk}}$ is still significantly less efficient than VRFs in the ROM. Moreover, by now there are only three VRFs that are secure under standard assumptions in the standard model [HJ16, Koh19, Ros18]. Even though Kohl's VRF achieves very short proofs of only $\omega(1)$ many group elements, this comes at the cost of larger verification keys and still does not provide as much efficiency as the VRFs in the ROM.

**Research Question 2.** *Are there practically efficient VRFs that provide post-quantum security?*

All VRFs that we discussed so far and in particular those about to be standardized [GRPV21] do not provide post-quantum security. That is, the complexity assumptions underlying these constructions are known to be solvable in polynomial time by quantum computers once they become available in sufficiently large scale. Thus, VRFs providing security against quantum computers are a current topic of research and some constructions have been presented recently. Esgin *et al.* present an efficient VRF secure under the hardness of lattice problems in [EKS+20][5]. However, their VRF can only be evaluated a few times before leaking too much information about the secret evaluation key. Similarly, Leroux presents a VRF secure under isogeny assumptions [Ler21] that can only be evaluated once. Moreover, Buser *et al.* present two post-quantum secure VRFs from symmetric primitives in [BDE+21]. They first present a construction based on post-quantum secure *non-interactive zero-knowledge proof* (NIZK) and a PRF. This construction is rather efficient but has significantly larger proofs and takes significantly longer to evaluate and verify [BDE+21, Table 2] than the schemes that are about to be standardized [GRPV21]. The second construction by Buser *et al.* provides a competitive time to evaluate and verify but is stateful and takes a very long time to generate key pairs [BDE+21, Table 2].

While the limitations of these first post-quantum secure VRFs are acceptable for some applications, it is still desirable to construct post-quantum secure VRFs that have all the properties we discussed in Section 2.3.1.

[5]The publication by Esgin *et al.* [EKS+20] was presented at the "International Conference on Financial Cryptography and Data Security 2021". However, no peer-reviewed versions of the publications at this venue from 2021 are available at the time of writing.

# 4 Verifiable Random Functions with Optimal Tightness

All known standard model VRFs, include those that we introduced in Chapter 3, have a reduction loss that is much worse than what one would expect from known optimal constructions of closely related primitives like unique signatures. In this chapter, we show that:

1. Every security proof for a VRF that relies on a non-interactive assumption has to lose a factor of $Q$, where $Q$ is the number of adversarial queries. To that end, we extend the meta-reduction technique of Bader *et al.* [BJLS16] to also cover VRFs.

2. This raises the question: Is this bound optimal? We answer this question in the affirmative by presenting the first VRF with a reduction from the non-interactive $q$-DBDHI assumption to the security of VRF that achieves this optimal loss.

We thus paint a complete picture of the achievability of tight verifiable random functions: We show that a security loss of $Q$ is unavoidable and present the first construction achieving this bound.

Author's contributions. This chapter is based on [Nie21], which the author of this thesis is the single author of. Thus, all contributions in this chapter are due to the author.

## 4.1 Motivation

Following the reductionist approach to security, we relate the difficulty of breaking the security of a cryptographic scheme to the difficulty of solving an underlying hard problem. Let $\lambda$ be the security parameter and consider a reduction showing that any adversary that breaks the security of a cryptographic scheme in time $t(\lambda)$ with probability $\varepsilon(\lambda)$ implies an algorithm that solves the underlying hard problem with probability $\varepsilon'(\lambda)$ in time $t'(\lambda)$ with $t'(\lambda) \geq t(\lambda)$ and $\varepsilon'(\lambda) \leq \varepsilon(\lambda)$. We then say that the reduction *loses* a factor $\ell(\lambda)$ if

$$\frac{t'(\lambda)}{\varepsilon'(\lambda)} \geq \frac{\ell(\lambda)t(\lambda)}{\varepsilon(\lambda)}$$

for all $\lambda \in \mathbb{N}$. We say that a reduction is *tight* if $\ell$ is a constant, *i.e.*, if the quality of the reduction does not depend on the security parameter.

The loss of a reduction is of particular practical importance when deciding on the key sizes to use for cryptographic schemes. For simplicity, assume that we have a reduction with $\varepsilon'(\lambda) = \varepsilon(\lambda)$ and $t'(\lambda) = \ell(\lambda)t(\lambda)$ and let $t_{\mathsf{opt}}(\lambda)$ denote the time the fastest algorithm takes to solve an instance of the complexity assumption. Then, if we want to rule out the existence of an adversary that breaks the scheme's security faster than $t_{\mathsf{adv}}$, we have to choose the security parameter large enough such that $t_{\mathsf{opt}}(\lambda)/\ell(\lambda) \geq t_{\mathsf{adv}}$. Hence, if $\ell$ is large, then $\lambda$ has to be rather large to guarantee that any adversary that breaks the security of the scheme has runtime at least $t_{\mathsf{adv}}$. However, a large security parameter also implies large keys, which negatively affects the real-world efficiency of the scheme. On the positive side, this means that if we are able to construct a tight reduction, this allows us to use *small key sizes and guarantee security* against all adversaries with runtime at most $t_{\mathsf{adv}}$. This approach to security is also known as concrete security and is more thoroughly discussed in [BR09b].

Impossibility of tight reductions. Unfortunately, we know that tight reductions can not exist for some primitives. Coron presented the first result of this kind in 2002 for unique signatures [Cor02], in which he showed that every security reduction for unique signatures loses at least a factor of $\approx Q$, where $Q$ is the number of adaptive signature queries made by the forger. He achieved this result by introducing the *meta-reduction technique*. That is, one shows that a tight reduction can not exist by proving that any tight reduction would solve the underlying hard problem without the help of an adversary. Note that the full proof is only contained in the full version [Cor01]. Subsequently, the technique has been successfully used to prove the same lower bound for the loss of security reductions for efficiently re-randomizable signatures by Hofheinz *et al.* [HJK12] and later on to an even broader class of primitives by Bader *et al.* [BJLS16]. Most recently, Coron's technique has been extended by further publications. First, Morgan and Pass extended Coron's technique to incorporate interactive complexity assumptions and reductions that execute several instances of an adversary in parallel. However, since the result applies to a broader class of reductions and complexity assumptions, the lower bound on the loss is only $\sqrt{Q}$ instead of $Q$. Moreover, Morgan *et al.* applied the technique to MACs and PRFs [MPS20].

Even though VRFs are closely related to unique signatures, none of the lower bounds on the loss mentioned above applies to VRFs in general because the non-interactive proofs of VRFs do not need to be unique, nor do they need to be re-randomizable. For example, the VRF by Bitansky does not have unique proofs of correctness [Bit20]. Hence, in contrast to a remark in [MP18], a VRF does not immediately imply a unique signature scheme but just a signature scheme with a unique component in its signatures.

Circumventing tightness lower bounds. Despite all the lower bounds on the loss of reductions to the security of unique signatures, Guo *et al.* showed in [GCS+17] that reductions circumventing the lower bounds are possible by making heavy use of the programmability of a random oracle. However, this technique is only applicable in the random oracle model and can not be adapted in the standard model to the best of our knowledge.

Moreover, the tightness lower bounds have also been circumvented in the standard model by making the signatures non-re-randomizable [AFLT12, BKKP15, CD96, HJ12, KW03, Sch11]. Kakvi and Kiltz even describe a tightly secure unique signature scheme by using a public key in the reduction that allows for non-unique signatures and is indistinguishable from an honestly generated public key [KK12, KK18].

Furthermore, for identity-based encryption schemes, which are closely related to VRFs [ACF14], Chen and Wee [CW13], and also Blazy *et al.* [BKP14] in parallel, describe a scheme that can be proven secure with a reduction whose loss depends only on the security parameter and not on the number of queries made by the adversary. In 2016, Boyen and Li then presented the first tightly secure IBE scheme in [BL16]. Similar to our approach in this work, they homomorphically evaluate a pseudorandom function in the reduction. However, they use it in order to apply the technique of Katz and Wang to construct tightly secure signatures by making the signatures non-re-randomizable [KW03].

However, the techniques above do not apply to VRFs. Replacing the verification with an indistinguishable verification key that allows for non-unique signatures is not possible due to the strong uniqueness requirement. Moreover, our meta reduction makes no assumptions about the re-randomizability of the proof of correctness produced by a VRF evaluation. Hence, making the proofs of correct evaluation non-re-randomizable can not allow for tighter reductions. Thus, to the best of our knowledge, the only avenues to achieve tighter reductions for VRFs would be to either use the random oracle model, prove the security from an interactive assumption, or use a reduction that can run several instances of an adversary in parallel. However, it seems unlikely to achieve a loss better than $\sqrt{Q}$ through the latter two approaches due to the lower bound by Morgan and Pass [MP18].

Our contributions. In this chapter, we study the tightness of reductions from non-interactive complexity assumptions to the security of verifiable random functions.

1. We first extend the lower bound for the loss of re-randomizable signatures from Bader *et al.* [BJLS16] to *verifiable unpredictable function*s (VUFs), which differ from VRFs in that the output only has to be unpredictable instead of pseudorandom. Since this is a weaker requirement, the theorem for VUFs also implies the same bound for reductions to the security of VRFs. Concretely, we prove that any reduction from a non-interactive complexity assumption to the unpredictability of a VUF loses a factor of at least $Q$.

2. We present a VRF and a reduction from the non-interactive $q$-DBDHI assumption to the adaptive pseudorandomness of the VRF that achieves this bound. The VRF is based on a VRF by Yamada in [Yam17b, Appendix C], which is the full version of [Yam17a].

## 4.2 Technical Overview

Before presenting our results, we give a short overview of our techniques below. We first describe how we prove the lower bound for the loss of VRFs and then describe our construction attaining this bound. Furthermore, recall that we model probabilistic algorithms in Definition 4 as TMs with a designated input tape on which the TM receives uniformly random input bits $\rho$. Moreover, we write $\mathcal{A}(x;\rho)$ to execute $\mathcal{A}$ on input $x$ with random bits $\rho$ and that we view $\mathcal{A}$ as deterministic if $\rho$ is explicitly specified. We recall these definitions here because we will make heavy use of executing probabilistic algorithms with specific randomness $\rho$.

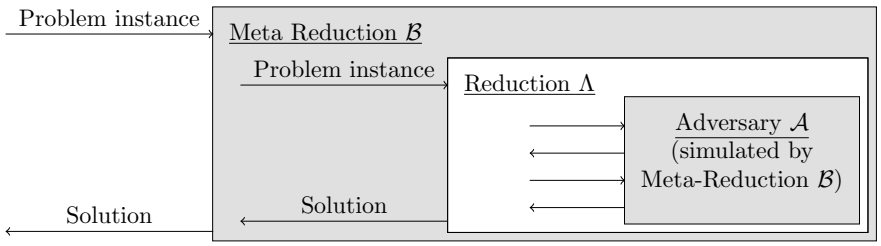


*Figure 4.1:* The meta-reduction technique of Coron [Cor02].

Bounding the tightness of VRFs. The results by Bader *et al.* apply to *re-randomizable* signatures/relations[1], which is a large call of public-key primitives, the do not cover VRFs and VUFs. This is because the non-interactive proofs of correctness given by VRFs and VUFs are not required to be re-randomizable. Since, to the best of our knowledge, all previous constructions of VRFs and VUFs have unique or re-randomizable proofs of correctness, this opens a gap for VRFs and VUFs with potentially tighter security proofs. Therefore, we extend the meta-reduction of Bader *et al.* to VRFs and VUFs and thus show that any reduction from a non-interactive complexity assumption to the security of a VRF necessarily loses a factor of at least $Q$, where $Q$ is the number of queries made by the adversary. This closes the gap left by previous meta-reductions. In order to explain how we extend their technique, we shortly revisit Coron's meta-reduction technique depicted in Figure 4.1. A *meta-reduction* can be thought of as a reduction against a reduction. That is, the meta-reduction $\mathcal{B}$ simulates a hypothetical adversary $\mathcal{A}$ for a reduction

[1]Note that unique signatures are re-randomizable because, given a unique signature for a message, it is trivial to sample from all signatures for that message since there is only that one signature.

$\Lambda$. Since the meta-reduction is constructed to have a polynomial runtime and simulates the hypothetical adversary, it is the reduction $\Lambda$ that solves the instance of the complexity assumption. This allows us to show that any reduction with a certain tightness can break the underlying complexity assumption without the help of any adversary and therefore contradicts the complexity assumption.

In their proof, Bader *et al.* use the re-randomizability/uniqueness of the signatures that $\Lambda$ produces for $\mathcal{A}$ to solve the challenge when simulating $\mathcal{A}$. We extend their technique to VRFs/VUFs by showing that it is sufficient if the part of the signature that the adversary has to provide for the challenge, in the case of VUFs, the unpredictable value $Y$, is unique or re-randomizable.

For simplicity, we prove the theorem for VUFs: this automatically implies the same bound for VRFs because every VRF is also a VUF. Following Bader *et al.*, we consider a very weak security model in which the number of queries $Q$ is fixed a priori. Further, the adversary is presented with $Q$ uniformly random and pairwise distinct inputs $X^{(1)}, \ldots, X^{(Q)}$ and has to choose a challenge $X^*$ from these. For all other inputs, the adversary is then given the VUF output and proof. Finally, the adversary has to output the VUF value for the challenge input and wins if the output is correct. We refer to this very weak security notion as *weak-selective unpredictability*. We describe a hypothetical adversary that breaks the weak-selective unpredictability with certainty and then show that our meta-reduction can efficiently simulate this adversary for the reduction. Informally, on input a problem instance for a non-interactive complexity assumption, the meta-reduction $\Lambda$ behaves as follows.

1. It passes on the problem instance to the reduction and lets it output a verification key $\mathsf{vk}$ and $Q$ pairwise different VUF inputs $X^{(1)}, \ldots, X^{(Q)}$.

2. It then iterates over all $j \in [Q]$ and executes the second part of the reduction as if it chose $j$ as the challenge and lets the reduction produce all pairs of VUF output and proof except for the $j$th pair. It then verifies them and saves them if they are correct with respect to $\mathsf{vk}$ and the corresponding input.

3. Finally, it chooses $j^* \xleftarrow{\$} [Q]$ and passes on the correct VUF output for $X^{(j^*)}$ to the reduction. We formally prove in Section 4.3 that the meta-reduction indeed has learned the correct VUF output for $X^{(j^*)}$ from the reduction with probability at least $1/Q$.

4. When the reduction then outputs the solution to the underlying problem instance, the meta-reduction outputs this solution as well.

Overall, we can then show that the meta-reduction takes time at most $\mathcal{B} = Q \cdot t_\Lambda + Q(Q+1)t_{\mathsf{Vfy}}$ and has a success probability of at least $\varepsilon_\Lambda - 1/Q$, where $t_\Lambda$ and $\varepsilon_\Lambda$ are the runtime and the success probability of the reduction and $t_{\mathsf{Vfy}}$ is the time it takes to verify a VUF output. Now, we conclude that $\Lambda$ has a loss of at least $\ell = (\varepsilon_N + 1/Q)^{-1}$, where $\varepsilon_N$ is the largest probability any algorithm running in time

| Schemes | Security loss |
|---|---|
| Hohenberger and Waters [HW10] | $\mathcal{O}(\lambda Q/\varepsilon)$ |
| Boneh *et al.* Sec. 7 in [BMR10b] | $(Q\lambda)^{\tau(\varepsilon)}$ |
| Jager [Jag15] | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| Hofheinz and Jager [HJ16] | $\mathcal{O}(\lambda \log(\lambda) Q^{2/c}/\varepsilon^3)$ |
| Yamada Sec 6.1 in [Yam17a] | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| Yamada Sec. 6.2 in [Yam17a] | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| Yamada App. C in [Yam17b] | $\mathcal{O}(\lambda^2 Q/\varepsilon^2)$ |
| Katsumata Sec. 5.1 in [Kat17] | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| Kastumata Sec. 5.3 in [Kat17] | $\mathcal{O}(Q^\nu/\varepsilon^{\nu+1})$ |
| Roşie [Ros18] | $\mathcal{O}(\lambda \log(\lambda) Q^{2/c}/\varepsilon^3)$ |
| Kohl [Koh19] | $\mathcal{O}(\lvert\pi\rvert \log(\lambda) Q^{2/\nu}/\varepsilon^3)$ |
| Kohl [Koh19] | $\mathcal{O}(\lvert\pi\rvert \log(\lambda) Q^{2+2/\nu}/\varepsilon^3)$ |
| Jager and Niehues [JN19a] | $\mathcal{O}(t^2/\varepsilon)$ |
| Jager *et al.* [JKN21] | $\mathcal{O}(t^2/\varepsilon)$ |
| Section 4.4.2 | $\mathcal{O}(Q)$ |

*Table 4.1:* We compare the loss of previous VRFs with all desired properties. For the variables, let $|\pi|$ denotes the size of the proofs of the VRF and $\varepsilon, t$ and $Q$ the advantage, runtime and number of queries made by the adversary the reduction is run against. Further, there are three values that depend on the error correcting code used in the construction: the function $\tau(\varepsilon) > 1$ and the constants $\nu > 1$ and $c \leq 1/2$. Note that the full version [BMR10a] of [BMR10b] has been updated with the bound stated above.

$t_{\mathcal{B}}$ has in breaking the complexity assumption. Since the complexity assumption implies that $\varepsilon_N$ is negligibly small, we have that $\ell \approx Q$.

While the meta-reduction above is only applicable to reductions that execute the adversary exactly once, our proof of the lower bound on the loss of VRFs in Section 4.3, like the one by Bader *et al.*, also applies to reductions that can sequentially rewind the adversary.

On the difficulty of constructing tightly secure VRFs. We recall the overview of constructions of VRFs with security in the standard model that we gave in Table 3.1 in Chapter 3, focusing on tightness in Table 4.1. It shows that known security proofs for VRF in the standard model are significantly more lossy than the lower bound $Q$. This situation raises the question:

*Do verifiable random functions with a loss of $Q$ exist?*

In consequence, such a VRF would show that a loss of $Q$ is indeed optimal.

We proceed to explain why all previous constructions have a much worse loss than $Q$ and then give an overview of our approach that achieves the optimal tightness.

To the best of our knowledge, the security proofs for all VRFs in the standard model are partitioning arguments as we discussed them in Section 2.6. That is, the reduction makes a guess in the very beginning and then has to abort and output a random bit depending on the queries and the challenge of the adversary. As we discussed in Section 2.6, such a reduction strategy requires upper and lower bounds on the probability to abort that are close to each other to prove the security of a VRF. Proving such upper and lower bounds is non-trivial because the probability to abort may depend on the queries and the challenge chosen by the adversary. The two most notable techniques addressing this requirement for VRFs are the artificial abort technique by Waters [Wat05] and the balancing technique by Bellare and Ristenpart [BR09a], which we discuss in detail in Section 2.6. Nonetheless, in conclusion, none of the techniques known so far achieves the optimal loss of $Q$.

A reduction with optimal tightness. We next answer the question stated above in the affirmative by presenting a VRF with a reduction that only loses a factor of $Q$. To do so, we have to address the issue raised above: that the success probability for the partitioning argument depends on the sequence of queries made by the adversary. We achieve this by passing every query and the challenge of the adversary through a pseudorandom function (PRF). Further, we utilize a property of the VRF Yamada introduced in [Yam17b, Appendix C]. This VRF allows the reduction to homomorphically embed an arbitrary NAND circuit of polynomial size and logarithmic depth in the VRF. The idea here is that the reduction can embed an arbitrary NAND-circuit in the VRF such that it can answer all queries by the adversary for which the circuit evaluates to 0 and can extract a solution to the underlying hard problem whenever the circuit evaluates to 1. In particular, the homomorphic evaluation hides selected bits of the circuit's inputs, all internal state bits of the circuit, and the circuit's output from the adversary.

We use these properties to evaluate a PRF homomorphically. Since the adversary does not learn any internal states or outputs of the PRF, we have that the outputs of the PRF are distributed as if they were the outputs of a random function. In particular, we then have that the outputs of the PRF are distributed uniformly and independent of each other. We show in Section 4.4.1 that it then suffices for the reduction to guess $\lceil \log(Q) \rceil + 1$ bits of the PRF output for the challenge input. Then the probability that the following two events both occur is at least $1/8Q$:

1. The PRF output of the challenge matches the guess.

2. The guess does not match the PRF output for any of the adversary's queries.

Further, viewing the PRF outputs as the output of a truly random function, the probability for the reduction to succeeds is independent of the probability of the adversary breaking the security of the VRF. Ultimately, this yields a VRF, which has a loss of $Q$ plus the loss of the PRF.

$G^{\mathsf{Unpred}}_{(\mathcal{A}_1,\mathcal{A}_2),\mathcal{VUF}}(\lambda)$

$(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVUF}(1^\lambda); \rho_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^\lambda$
$(X^*, Y^*) := \mathcal{A}^{\mathsf{Eval}(\mathsf{sk},\cdot)}(\mathsf{vk}; \rho_{\mathcal{A}})$
$(Y, \pi) \xleftarrow{\$} \mathsf{Eval}(\mathsf{sk}, X^*)$
**if** $Y == Y^*$
  **return** 1
**else**
  **return** 0

*Figure 4.2:* The security experiment specifying unpredictability.

## 4.3 Impossibility of VUFs and VRFs with Tight Reductions

In this section, we prove that any reduction from a non-interactive complexity assumption to the security of a VUF or VRF unavoidably loses a factor of $Q$. To that end, we formally introduce VUFs and their accompanying security notion. We then introduce a very weak security notion for VUFs and prove that even for this notion, every reduction form a non-interactive complexity assumption to it necessarily loses a factor of $Q$.

### 4.3.1 Verifiable Unpredictable Functions

Syntactically, a *verifiable unpredictable function* (VUF) consists of the same algorithms as a VRF. However, for a VUF, we require the weaker security notion of *unpredictability* instead of pseudorandomness. That is, instead of providing a challenge input $X^*$, we require the adversary to provide a challenge input $X^*$ together with the output $Y^*$ of the VUF for $X^*$. We formalize these properties below.

**Definition 22.** A *verifiable unpredictable function* (VUF) with domain $\mathcal{X}$ and finite range $\mathcal{Y}$ consists of three algorithms $\mathcal{VUF} = (\mathsf{SetupVUF}, \mathsf{Eval}, \mathsf{Vfy})$ with the following syntax.

- $(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVUF}(1^\lambda)$ takes as input the security parameter $\lambda$ and outputs a key pair $(\mathsf{vk},\mathsf{sk})$. We say that $\mathsf{sk}$ is the *secret key* and $\mathsf{vk}$ is the *verification key*.
- $(Y,\pi) \xleftarrow{\$} \mathsf{Eval}(\mathsf{sk}, X)$ takes as input a secret key $\mathsf{sk}$ and an input $X \in \mathcal{X}$, and outputs a function value $Y \in \mathcal{Y}$ and a proof $\pi$.
- $\mathsf{Vfy}(\mathsf{vk}, X, Y, \pi) \in \{0,1\}$ takes as input a verification key $\mathsf{vk}$, $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, and proof $\pi$, and outputs a bit.

$G^{Q\text{-ws-Unpred}}_{(\mathcal{A}_1,\mathcal{A}_2),\mathcal{VUF}}(\lambda)$

$(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}(1^\lambda); \rho_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^\lambda$

$(X^{(1)},\ldots,X^{(Q)}) \xleftarrow{\$} \mathcal{X}$ s.t. $X^{(i)} \neq X^{(j)}$ for all $i \neq j$

$(Y_i,\pi_i) \xleftarrow{\$} \mathsf{Eval}(\mathsf{sk}, X^{(i)})$

$(j,\mathsf{st}) := \mathcal{A}_1(\mathsf{vk}, (X^{(i)})_{i\in[Q]}; \rho_{\mathcal{A}})$

$Y^* := \mathcal{A}_2((Y_i,\pi_i,\mathsf{st})_{i\in[Q\setminus j]})$

**return** $Y^* == Y_j$

*Figure 4.3:* The security experiment specifying weak selective unpredictability.

We say that $\mathcal{VUF} = (\mathsf{SetupVUF}, \mathsf{Eval}, \mathsf{Vfy})$ with domain $\mathcal{X}$ and range $\mathcal{Y}$ is a secure VUF if it fulfills the following requirements.

**Correctness.** We say that $\mathcal{VUF}$ has *correctness* if for all $(\mathsf{vk},\mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVUF}(1^\lambda)$, $X \in \mathcal{X}$ and, $(Y,\pi) \xleftarrow{\$} \mathsf{Eval}(\mathsf{sk}, X)$ it must hold that $\mathsf{Vfy}(\mathsf{vk}, X, Y, \pi) = 1$. Further, the algorithms $\mathsf{SetupVUF}$, $\mathsf{Eval}$, $\mathsf{Vfy}$ have to be PPTs.

**Unique provability.** We say that $\mathcal{VUF}$ has *unique provability* if for all $\mathsf{vk} \in \{0,1\}^*$ and all $X \in \mathcal{X}$, there does not *exist* any $Y_0, \pi_0, Y_1, \pi_1 \in \{0,1\}^*$ such that $Y_0 \neq Y_1$, and it holds that $\mathsf{Vfy}(\mathsf{vk}, X, Y_0, \pi_0) = \mathsf{Vfy}(\mathsf{vk}, X, Y_1, \pi_1) = 1$.

**Unpredictability.** Consider an adversary $\mathcal{A}$ with access (via oracle queries) to $\mathsf{Eval}(\mathsf{sk}, \cdot)$ in the unpredictability game depicted in Figure 4.2. We say that $\mathcal{A}$ is *legitimate* if there is no $\rho_{\mathcal{A}} \in \{0,1\}^\lambda$ such that $\mathcal{A}$ query $\mathsf{Eval}(\mathsf{sk}, X^*)$, where $X^* \in \mathcal{X}$ is part of the output of $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in breaking the unpredictability of $\mathcal{VUF}$ as

$$\mathsf{Adv}^{\mathsf{Unpred}}_{\mathcal{VUF},\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{Unpred}}_{\mathcal{VUF},\mathcal{A}}(\lambda) = 1\right]$$

and say that $\mathcal{VUF}$ is *unpredictable* if for every PPT adversary $\mathcal{A}$ it holds that $\mathsf{Adv}^{\mathsf{Unpred}}_{\mathcal{VUF},\mathcal{A}}(\lambda)$ is negligible in $\lambda$.

We note that every secure VRF is also a secure VUF. This is because an adversary that can compute the correct output of a VRF, *i.e.*, can predict it, can trivially distinguish the correct output from a random element of the range of the VRF.

### 4.3.2 Lower Tightness Bounds for VUFs

We begin by introducing the very weak security notion of *weak-selective unpredictability* for VUFs. In this security model, all queries and the challenge are uniformly random and pairwise distinct. We formally define it as follows.

**Definition 23.** Let $\mathcal{VUF} = (\mathsf{SetupVUF}, \mathsf{Eval}, \mathsf{Vfy})$ be a verifiable unpredictable function and let $t : \mathbb{N} \to \mathbb{N}, \varepsilon : \mathbb{N} \to [0,1]$. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we

$\mathsf{NICA}^N_{\mathcal{A}}(\lambda)$

$(c, w) \xleftarrow{\$} \mathsf{T}(1^\lambda); \rho_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^\lambda$

$s \coloneqq \mathcal{A}(c; \rho_{\mathcal{A}})$

**return** $\mathsf{V}(c, w, s)$

*Figure 4.4:* The generic security experiment for a non-interactive complexity assumption $N = (\mathsf{T}, \mathsf{V}, \mathsf{U})$ between the challenger and an adversary $\mathcal{A}$.

say that $\mathcal{A}$ $(t, Q, \varepsilon)$-breaks the weak selective unpredictability of $\mathcal{VUF}$ if $\mathcal{A}$ runs in time $t$ and

$$\mathsf{Adv}^{\mathsf{ws\text{-}Unpred}}_{\mathcal{VUF},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) \coloneqq \Pr\left[G^{\mathsf{ws\text{-}Unpred}}_{\mathcal{VUF},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1\right] = \varepsilon(\lambda)$$

where $G^{\mathsf{ws\text{-}Unpred}}_{\mathcal{VUF},(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$ is the security experiment depicted in Figure 4.3.

Note that any VRF fulfilling the requirements of Definition 6 and every VUF fulfilling the requirements of Definition 22 has also weak-selective unpredictability. Hence, ruling out a tight reduction from weak selective unpredictability to a class of complexity assumptions, also rules out tight reductions from pseudorandomness and unpredictability to that class of complexity assumptions. We thus prove a lower bound on the loss of any reduction from any non-interactive complexity assumption to the weak selective unpredictability of a VUF, where the reduction my sequentially repeat the execution of the adversary.

To be more precise in our definitions, we follows [AGO11, BJLS16], and explicitly state that the algorithms we consider are *Turing machine*s (TMs) as defined in Section 2.2. We define a non-interactive complexity assumption as a triple $N = (\mathsf{T}, \mathsf{V}, \mathsf{U})$ of TMs. While the TM $\mathsf{T}$ generates a problem instance and $\mathsf{V}$ verifies the correctness of a solution, the TM $\mathsf{U}$ represents a trivial adversary to compare an actual adversary against. For example, a trivial adversary against the DDH assumption would just output random bit as its guess. We formally define non-interactive complexity assumptions as follows.

**Definition 24.** A *non-interactive complexity assumption* $N = (\mathsf{T}, \mathsf{V}, \mathsf{U})$ consist of three Turing machines. The instance generation machine $(c, w) \xleftarrow{\$} \mathsf{T}(1^\lambda)$ takes the security parameter as input and outputs a problem instance $c$ and a witness $w$. $\mathsf{U}$ is a probabilistic polynomial-time Turing machine, which takes $c$ as input and outputs a candidate solution $s$. The verification Turing machine $\mathsf{V}$ takes as input $(c, w)$ and a candidate solution $s$. If $\mathsf{V}(c, w, s) = 1$, then we say that $s$ is a correct solution to the challenge $c$.

*Remark* 8. We note that all complexity assumptions used in this thesis, and in particular those we introduced in Section 2.4, fall in the category of non-interactive complexity assumptions.

$r\text{-}\Lambda^{\mathcal{A}}(c, \rho_\Lambda)$

$\mathsf{st}_{\Lambda_{1,1}} \coloneqq \Lambda_1(c; \rho_0)$
**for** $1 \leq \ell \leq r$ :
  $(\mathsf{vk}^\ell, (X^{(i)^\ell})_{i\in[Q]}, \rho_{\mathcal{A}}, \mathsf{st}_{\Lambda_{\ell,2}}) \coloneqq \Lambda_{\ell,1}(\mathsf{st}_{\Lambda,1})$
  $(j^{*\ell}, \mathsf{st}_{\mathcal{A}}) \coloneqq \mathcal{A}_1(\mathsf{vk}^\ell, (X^{(i)^\ell})_{i\in[Q]}; \rho_{\mathcal{A}})$
  $((Y_i^\ell, \pi_i^\ell)_{i\in[Q\setminus j^{*\ell}]}, \mathsf{st}_{\Lambda_{\ell,3}}) \coloneqq \Lambda_{\ell,2}(j^{*\ell}, \mathsf{st}_{\Lambda_{\ell,2}})$
  $Y_{j^{*\ell}}^\ell \coloneqq \mathcal{A}_2((Y_i^\ell, \pi_i^\ell)_{i\in[Q\setminus j^{*\ell}]}, \mathsf{st}_{\mathcal{A}})$
  $\mathsf{st}_{\Lambda_{\ell+1,1}} \coloneqq \Lambda_{\ell,3}\left(Y_{j^{*\ell}}^\ell, j^{*\ell}, \mathsf{st}_{\Lambda_{\ell,3}}\right)$
$s \coloneqq \Lambda_3(\mathsf{st}_{\Lambda r+1,1})$

*Figure 4.5:* Description of the Turing machine $r\text{-}\Lambda^{\mathcal{A}}$ built from an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the weak selective unpredictability of a verifiable unpredictable function and a reduction $(\Lambda_1, (\Lambda_{\ell,1}, \Lambda_{\ell,2}, \Lambda_{\ell,3})_{\ell\in[r]}, \Lambda_3)$.

**Definition 25.** Let $N = (\mathsf{T}, \mathsf{V}, \mathsf{U})$ be a non-interactive complexity assumption and let $\mathsf{NICA}$ be the security experiment depicted in Figure 4.4. For functions $t : \mathbb{N} \to \mathbb{N}, \varepsilon : \mathbb{N} \to [0, 1]$ and a probabilistic Turing machine $\mathcal{B}$ running in time $t(\lambda)$, we say that $\mathcal{B}$ $(t, \varepsilon)$-breaks $N$ if

$$\left|\Pr\left[\mathsf{NICA}_{\mathcal{B}}^N(\lambda) = 1\right] - \Pr\left[\mathsf{NICA}_{\mathsf{U}}^N(\lambda) = 1\right]\right| \geq \varepsilon(\lambda),$$

where the probabilities are taken over the randomness consumed by $\mathsf{T}$ and the random choices of $\rho_{\mathsf{U}}$ and $\rho_{\mathcal{B}}$ in the security experiments $\mathsf{NICA}_{\mathcal{B}}^n(\lambda)$ and $\mathsf{NICA}_{\mathsf{U}}^n(\lambda)$.

Bader *et al.* prove lower bounds for simple reductions as well as for reductions that can sequentially rewind the adversary [BJLS16]. Since the latter class of reductions include the former class, we directly prove the lower bound on the loss for the larger class of reductions. Following Bader *et al.*, we view a reduction that sequentially rewinds an adversary up to $r \in \mathbb{N}$ times as a $3r + 2$-tuple of Turing machines. That is, one TM that initializes the reduction, one to produce a solution in the end and three for each execution of the adversary. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the weak selective unpredictability of a verifiable unpredictable function $\mathcal{VUF}$, we let $r\text{-}\Lambda^{\mathcal{A}}$ be the Turing machine depicted in Figure 4.5.

**Definition 26** (Def. 6 in [BJLS16])**.** For a verifiable unpredictable function $\mathcal{VUF}$, we say that a Turing machine $r\text{-}\Lambda = (\Lambda_1, (\Lambda_{\ell,1}, \Lambda_{\ell,2}, \Lambda_{\ell,3})_{\ell\in[r]}, \Lambda_3)$ is an $r$-simple $(t_\Lambda, Q, \varepsilon_\Lambda, \varepsilon_{\mathcal{A}})$-reduction from breaking the non-interactive complexity assumption $N = (\mathsf{T}, \mathsf{V}, \mathsf{U})$ to breaking the weak selective unpredictability of $\mathcal{VUF}$ if for any TM $\mathcal{A}$ that $(t_{\mathcal{A}}, Q, \varepsilon_{\mathcal{A}})$-breaks the weak selective unpredictability of $\mathcal{VUF}$, TM $r\text{-}\Lambda^{\mathcal{A}}$ as defined in Figure 4.5 $(t_\Lambda + rt_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ breaks $N$.

Furthermore, we define the loss of a reduction as the factor that the quotient $(t_\Lambda(\lambda)+rt_{\mathcal{A}}(\lambda))/\varepsilon_\Lambda(\lambda)$ is larger than $t_{\mathcal{A}}(\lambda)/\varepsilon_{\mathcal{A}}(\lambda)$. We formalize this in the following definition.

**Definition 27.** For a verifiable unpredictable function $\mathcal{VUF}$, a non-interactive complexity assumption $N$, a function $\ell : \mathbb{N} \to \mathbb{N}$ and a reduction $\Lambda$, we say that $\Lambda$ loses $\ell$, if there exists an adversary $\mathcal{A}$ that $(t_{\mathcal{A}}, Q, \varepsilon_{\mathcal{A}})$ breaks the weak selective unpredictability of $\mathcal{VUF}$ such that $\Lambda^{\mathcal{A}}$ $(t_\Lambda + r \cdot t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks $N$, where

$$\frac{t_\Lambda(\lambda) + rt_{\mathcal{A}}(\lambda)}{\varepsilon_\Lambda(\lambda)} \geq \ell(\lambda) \cdot \frac{t_{\mathcal{A}}(\lambda)}{\varepsilon_{\mathcal{A}}(\lambda)}.$$

After introducing the needed notations and notions, we can now state our theorem regarding the loss of VRFs and VUFs.

**Theorem 7.** *Let $N = (\mathsf{T}, \mathsf{V}, \mathsf{U})$ be a non-interactive complexity assumption, $Q, r \in \mathsf{poly}(\lambda)$ and let $\mathcal{VUF}$ be a verifiable unpredictable function. Then for any $r$-simple $(t_\Lambda, Q, \varepsilon_\Lambda, 1)$-reduction $\Lambda$ from breaking $N$ to breaking the weak selective unpredictability of $\mathcal{VUF}$ there exists a TM $\mathcal{B}$ that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}})$-breaks $N$, where*

$$t_{\mathcal{B}} \leq r \cdot Q \cdot t_{\mathcal{A}} + r \cdot Q \cdot (Q-1) \cdot t_{\mathsf{Vfy}}$$
$$\varepsilon_{\mathcal{B}} \geq \varepsilon_\Lambda - \frac{r}{Q}.$$

*Here, $t_{\mathsf{Vfy}}$ is time needed to run the algorithm $\mathsf{Vfy}$ of $\mathcal{VUF}$.*

Note that the theorem also applies to adversaries with $\varepsilon_{\mathcal{A}} < 1$, as we discuss after the proof of Theorem 7. However, before proving Theorem 7, we show that it implies that every $r$-simple reduction $\Lambda$ from a non-interactive complexity assumption $N$ has at least a loss of $\approx Q$. For $t_N := t_{\mathcal{B}} = r \cdot Q \cdot t_\Lambda + r \cdot Q \cdot (Q-1) \cdot t_{\mathsf{Vfy}}$, let $\varepsilon_N$ be the largest probability such that there exists an algorithm that $(t_N, \varepsilon_N)$-breaks $N$. We then have that $\varepsilon_N \geq \varepsilon_{\mathcal{B}}$ and by Theorem 7, we have that $\varepsilon_\Lambda \leq \varepsilon_{\mathcal{B}} + r/Q \leq \varepsilon_N + r/Q$. We can then conclude that

$$\frac{t_\Lambda + r \cdot t_{\mathcal{A}}}{\varepsilon_\Lambda} \geq \frac{r \cdot t_{\mathcal{A}}}{\varepsilon_N + r/Q} = (\varepsilon_N + r/Q)^{-1} \cdot r \cdot \frac{t_{\mathcal{A}}}{1} = (\varepsilon_N + r/Q)^{-1} \cdot r \cdot \frac{t_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}}.$$

This means that $\Lambda$ loses at least a factor of $\ell = r/(\varepsilon_N + r/Q)$. Further, if $\varepsilon_N$ is very small, which it is supposed to be for a realistic complexity assumption, then $\ell \approx Q$.

*Proof.* Our proof is structured like the proofs in [BJLS16, HJK12, LW14] and thus first describes a hypothetical adversary that breaks the weak selective unpredictability of $\mathcal{VUF}$ with certainty and then describes a meta reduction that perfectly and efficiently simulates this adversary towards $\Lambda$.

THE HYPOTHETICAL ADVERSARY $\mathcal{A}$. The hypothetical adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ consists of the following two procedures.

$\mathcal{A}_1(\mathsf{vk}, (X^{(i)})_{i\in[Q]}; \rho_{\mathcal{A}})$ samples $j \xleftarrow{\$} [Q]$ and outputs $(j, \mathsf{st})$ with the state $\mathsf{st} = (\mathsf{vk}, (X^{(i)})_{i\in[Q]}, j)$.

$\mathcal{A}_2((Y_i, \pi_i)_{i\in[Q\setminus j]}, \mathsf{st})$ first parses the state $\mathsf{st}$ as $(\mathsf{vk}, (X^{(i)})_{i\in[Q]}, j)$ and checks whether $\mathsf{Vfy}(\mathsf{vk}, X^{(i)}, Y_i, \pi_i) = 1$ for all $i \in [Q \setminus j]$. If there is $i^*$ for which it holds that $\mathsf{Vfy}(\mathsf{vk}, X^{(i)}, Y_i, \pi_i) = 0$, it aborts with result $\perp$. Otherwise, it computes $Y^* \in \mathcal{Y}$ such that there exists $\pi \in \{0,1\}^*$ with $\mathsf{Vfy}(\mathsf{vk}, X^{(j)}, Y^*, \pi) = 1$. The existence and uniqueness of such a $Y^*$ is guaranteed by the correctness and the unique provability of $\mathcal{VUF}$.

Observe that $\mathcal{A}$ breaks the weak selective unpredictability of $\mathcal{VUF}$ with certainty because a correct VUF produces only valid pairs of outputs and proofs, but $\mathcal{A}_2$ may not be efficiently computable. However, we show that $\mathcal{B}$ can efficiently simulate $\mathcal{A}$ nonetheless.

THE META-REDUCTION $\mathcal{B}$. We now describe the meta-reduction $\mathcal{B}$ that simulates $\mathcal{A}$ $r$ times for the reduction $\Lambda = (\Lambda_1, (\Lambda_{\ell,1}, \Lambda_{\ell,2}, \Lambda_{\ell,3})_{\ell\in[r]}, \Lambda_3)$. $\mathcal{B}$'s goal in this is to break $N$ and is therefore called on input $c$, where $(c, w) \xleftarrow{\$} \mathsf{T}(1^\lambda)$.

i. $\mathcal{B}$ receives $c$ as input. It samples randomness $\rho_\Lambda \xleftarrow{\$} \{0,1\}^\lambda$ and executes $\mathsf{st}_{\Lambda_{1,1}} = \Lambda_1(c, \rho_\Lambda)$. If $\Lambda_1$ does not output $\mathsf{st}_{\Lambda_{1,1}}$, then $\mathcal{B}$ aborts and outputs $\perp$. Since the randomness of $\Lambda_1$ is fixed, we view all subroutines of $\Lambda$ as deterministic. Note that $\Lambda_1$ can pass on random coins to the other subroutines via $\mathsf{st}_{\Lambda_{1,1}}$.

ii. Next, $\mathcal{B}$ sequentially simulates $\mathcal{A}$ $r$ times for $\Lambda$. That is, for all $1 \le \ell \le r$ it does the following.

   a) Initialize an empty array $A^\ell$ with $Q$ places, that is $A^\ell[i] = \perp$ for all $i \in [Q]$.

   b) Run $(\mathsf{vk}^\ell, (X^{(i)^\ell})_{i\in[Q]}, \rho_{\mathcal{A}}, \mathsf{st}_{\Lambda_{\ell,2}}) = \Lambda_{\ell,1}(\mathsf{st}_{\Lambda_{\ell,1}})$. If $\Lambda_{\ell,1}$ does not produce such an output, then $\mathcal{B}$ aborts and outputs $\perp$.

   c) Then $\mathcal{B}$ runs $\left((Y^\ell_{i,j}, \pi^\ell_{i,j})_{i\in[Q\setminus j]}, \mathsf{st}_{\Lambda_{3,\ell}}\right) = \Lambda_{\ell,2}(j, \mathsf{st}_{\Lambda_{\ell,2}})$ for all $j \in [Q]$. If $\Lambda_{\ell,2}$ only produces correct outputs with respect to $\mathsf{vk}^\ell$, that is if

$$\bigwedge_{i\in[Q\setminus\ell]} \mathsf{Vfy}(\mathsf{vk}^\ell, X^{(i)^\ell}, Y^\ell_{i,j}, \pi^\ell_{i,j}) = 1$$

   holds, then $\mathcal{B}$ sets $A^\ell[i] \coloneqq Y^\ell_{i,j}$ for all $i \in [Q \setminus j]$.

   d) $\mathcal{B}$ then samples $j^{*\ell} \xleftarrow{\$} [Q]$. It then proceeds in one of the following cases:

      1. If $\Lambda_{\ell,2}(j^{*\ell}, \mathsf{st}_{\Lambda_{\ell,2}})$ produced any invalid pair of output and proof, that is, if there exists $i \in [Q \setminus j^{*\ell}]$ such that it holds that the $\mathsf{Vfy}$ rejects, that is $\mathsf{Vfy}(\mathsf{vk}^\ell, X^{(i)^\ell}, Y^\ell_{i,j^{*\ell}}, \pi^\ell_{i,j^{*\ell}}) = 0$, then $\mathcal{B}$ aborts and outputs $\perp$.

2. Otherwise, $\mathcal{B}$ sets $Y^* \coloneqq A^\ell[j^{*\ell}]$.

e) Set $\mathsf{st}_{\Lambda_{\ell+1,1}} \coloneqq \Lambda_{\ell,3}(Y^*, \mathsf{st}_{\Lambda_{\ell,3}})$

iii. Finally, $\mathcal{B}$ runs $s \xleftarrow{\$} \Lambda_3(\mathsf{st}_{\Lambda_{r+1,1}})$ and outputs $s$.

Success probability of $\mathcal{B}$. In order to analyze the success probability of $\mathcal{B}$, we compare the simulation of $\mathcal{A}$ by $\mathcal{B}$ with the description of $\mathcal{A}$. Note that $\mathcal{A}_1$ samples $j$ uniformly at random and $\mathcal{A}_2$ aborts if it is given an invalid pair of output and proof. $\mathcal{B}$ also samples $j^{*\ell}$ uniformly at random from $[Q]$ and aborts if $\Lambda_{\ell,2}(j^{*\ell}, \mathsf{st}_{\Lambda_{\ell,2}})$ produced any invalid pair of output and proof, just like $\mathcal{A}$. However, we are only guaranteed that $A^\ell[j^{*\ell}]$ contains the correct output of $\mathcal{VUF}$ for ${X^{(i)}}^\ell$ if there is $j' \in [Q \setminus j^{*\ell}]$ such that $\Lambda_{\ell,2}(j', \mathsf{st}_{\ell,2})$ outputs only correct pairs of outputs and proofs, *i.e.*, if this is not the case the simulation of $\mathcal{A}$ by $\mathcal{B}$ deviates from $\mathcal{A}$'s behavior. Below, we formally prove that $\mathcal{B}$ perfectly simulates $\mathcal{A}$ unless the event described above occurs and upper bound the probability that it occurs by $r/Q$.

Let $\mathsf{st}_{\Lambda_{\ell,2}}$ be the unique state computed by $\Lambda_{\ell,1}$ and let $j^{*\ell} \in [Q]$ be the unique index that $\Lambda_{\ell,3}$ is executed with. Note that these values are well-defined in both $\mathsf{NICA}_N^{\Lambda^{\mathcal{A}}}(\lambda)$ and $\mathsf{NICA}_N^{\mathcal{B}}(\lambda)$. Now, define the event $\mathsf{all\text{-}valid}(\mathsf{st}_{\Lambda_{\ell,2}}, j)$ as the event that $\Lambda_{\ell,2}$ outputs only valid pairs of outputs and proofs. That is

$$\mathsf{all\text{-}valid}(\mathsf{st}_{\Lambda_{\ell,2}}, j) \coloneqq \begin{cases} 1 & \text{if } \mathsf{Vfy}(\mathsf{vk}^\ell, {X^{(i)}}^\ell, Y_{i,j}^\ell, \pi_{i,j}^\ell) = 1 \text{ for all } i \in [Q \setminus j] \\ 0 & \text{otherwise,} \end{cases}$$

where $(Y_{i,j}^\ell, \pi_{i,j}^\ell)_{i \in [Q \setminus j]} = \Lambda_{\ell,2}(\mathsf{st}_{\Lambda_{\ell,2}}, j)$. Recalling the case in which $\mathcal{B}$'s simulation deviates the hypothetical adversary $\mathcal{A}$, we define the event

$$\mathsf{bad}(\ell) \coloneqq \mathsf{all\text{-}valid}(\mathsf{st}_{\Lambda_{\ell,2}}, j^{*\ell}) \bigwedge_{j \in [Q \setminus j^{*\ell}]} \neg \mathsf{all\text{-}valid}(\mathsf{st}_{\Lambda_{\ell,2}}, j).$$

Informally, the event $\mathsf{bad}(\ell)$ occurs if $\Lambda_{\ell,2}$ returned only valid pairs of outputs and proofs for $j = j^{*\ell}$ in the $\ell$'th simulation of $\mathcal{A}$. Further, we let $\mathsf{bad} \coloneqq \bigvee_{\ell \in [r]} \mathsf{bad}(\ell)$ be the event that $\mathsf{bad}(\ell)$ occurs for any $\ell \in [r]$.

Next, let $\mathsf{S}(\mathcal{F})$ denote the event that $\mathsf{NICA}_N^{\mathcal{F}}(\lambda) = 1$ for some adversary $\mathcal{F}$ against the non-interactive complexity assumption $N$. Then we observe the following:

$$\begin{aligned} &\Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}})\right] - \Pr\left[\mathsf{S}(\mathcal{B})\right] \\ &= \Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}}) \wedge \mathsf{bad}\right] + \Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}}) \wedge \neg\mathsf{bad}\right] - \Pr\left[\mathsf{S}(\mathcal{B}) \wedge \mathsf{bad}\right] - \Pr\left[\mathsf{S}(\mathcal{B}) \wedge \neg\mathsf{bad}\right] \\ &\leq \Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}}) \wedge \neg\mathsf{bad}\right] - \Pr\left[\mathsf{S}(\mathcal{B}) \wedge \neg\mathsf{bad}\right] + \Pr\left[\mathsf{bad}\right] \end{aligned}$$

Therefore, we proceed by showing two things:

1. $\Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}}) \wedge \neg\mathsf{bad}\right] = \Pr\left[\mathsf{S}(\mathcal{B}) \wedge \neg\mathsf{bad}\right]$
2. $\Pr\left[\mathsf{bad}\right] \leq r/Q$

In order to prove the first statement, we consider two cases in which $\mathcal{A}$ outputs either $\perp$ or the correct output of $\mathcal{VUF}$ for input $X^{(j)^\ell}$ under verification key $\mathsf{vk}^\ell$. These are the two cases that $\mathcal{B}$ distinguishes in step ii. d).

1. In the first case $\Lambda_{\ell,2}(j^{*\ell}, \mathsf{st}_{\Lambda_{\ell,2}})$ outputs $(Y^\ell_{i,j^{*\ell}}, \pi^\ell_{i,j^{*\ell}})_{i\in[Q\setminus j^{*\ell}]}$ such that there is $i \in [Q \setminus j^{*\ell}]$ with $\mathsf{Vfy}(\mathsf{vk}^\ell, X^{(i)^\ell}, Y^\ell_{i,j^{*\ell}}, \pi^\ell_{i,j^{*\ell}}) = 0$. Note that in this case, $\mathcal{A}_2$ aborts and outputs $\perp$. $\mathcal{B}$ also aborts and outputs $\perp$ in step ii. d) in the first case.

2. In the second case no such $i \in [Q \setminus j^{*\ell}]$ exists for the output of $\Lambda_{\ell,2}(j^{*\ell}, \mathsf{st}_{\Lambda_{\ell,2}})$. Hence, we have $\mathsf{all\text{-}valid}(\mathsf{st}_{\Lambda_{\ell,2}}, j^{*\ell}) = 1$. Furthermore, since we assumed that $\mathsf{bad}$ does not happen, we have that there is also $j \in [Q \setminus j^{*\ell}]$ with $\mathsf{all\text{-}valid}(\mathsf{st}_{\Lambda_{\ell,2}}, j) = 1$ and therefore $A^\ell[j^{*\ell}]$ contains the correct $\mathcal{VUF}$ output, which $\mathcal{B}$ passes on to $\Lambda_{\ell,3}$. Since $\mathcal{A}$ also outputs the correct output for $\mathcal{VUF}$ in this case, the two outputs are distributed identically.

Therefore, we have $\Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}}) \wedge \neg\mathsf{bad}\right] = \Pr\left[\mathsf{S}(\mathcal{B}) \wedge \neg\mathsf{bad}\right]$.

Next, we show that $\Pr[\mathsf{bad}] \leq r/Q$. For this, consider a fixed $\ell \in [r]$ and observe that $\mathsf{bad}(\ell)$ can occur only if there is a unique index $j \in [Q]$ such that $\mathsf{all\text{-}valid}(\mathsf{st}_{\ell,2}, j) = 1$ holds. Hence, the probability that $\mathcal{B}$ draws $j^{*\ell} = j$ in step ii. d) in the $\ell$'th round is $1/Q$. Therefore, we have that $\Pr[\mathsf{bad}(\ell)] = 1/Q$, and it follows by the union bound that $\Pr[\mathsf{bad}] \leq r/Q$. Summing up, we have shown that.

$$\Pr\left[\mathsf{S}(r\text{-}\Lambda^{\mathcal{A}})\right] - \Pr\left[\mathsf{S}(\mathcal{B})\right] \leq \Pr\left[\mathsf{bad}\right] \leq r/Q \iff \varepsilon_\Lambda \leq \varepsilon_{\mathcal{B}} - r/Q$$

It is now only left to compute the running time of $\mathcal{B}$. For this, note that $\mathcal{B}$ executes the algorithms $\Lambda_{\ell,2}$ $Q$ times for each $\ell \in [r]$ and other algorithms of $\Lambda$ only once. Furthermore, $\mathcal{B}$ executes $\mathsf{Vfy}$ $r \cdot Q \cdot (Q-1)$ times. Overall, we thus conclude that

$$t_{\mathcal{B}} \leq r \cdot Q \cdot t_\Lambda + r \cdot Q \cdot (Q-1) \cdot t_{\mathsf{Vfy}}$$

holds, where $t_{\mathsf{Vfy}}$ is the time it takes to execute $\mathsf{Vfy}$. This concludes the proof. □

Non-perfect adversaries. We only considered adversaries that always break the weak selective unpredictability of the VUF in the theorem above. However, the hypothetical adversary $\mathcal{A}$ and the meta-reduction can also simulate adversaries with arbitrary $\varepsilon_{\mathcal{A}} \in [0,1]$ by just aborting with probability $1-\varepsilon_{\mathcal{A}}$ in the simulation of $\mathcal{A}$.

## 4.4 Achieving Optimal Tightness for Verifiable Random Functions

Now that we showed that every reduction from a non-interactive complexity assumption to the pseudorandomness or unpredictability of a VRF or VUF, respectively, loses at least a factor of $Q$, we present a VRF together with a reduction,

which attains this bound up to a small constant factor. We achieve this by describing a *partitioning proof strategy*, which we discussed in Section 2.6. We first describe how the reduction chooses this partition in Section 4.4.1. We then discuss the embedding of the partitioning in the VRF in Section 4.4.2.

### 4.4.1 A Reduction Strategy with Optimal Tightness

In order to make a partitioning argument with optimal tightness for VRFs, we need to decouple the probability that the partitioning succeeds from the queries and the challenge, which are chosen by the adversary. We achieve this by passing every input of the adversary through a *pseudorandom function* (PRF). This ensures that the outputs are distributed independently and uniformly at random for pairwise different inputs. We formally define a PRF as below, similar to the definition in [KL14, Definition 3.25].

**Definition 28** (Pseudorandom functions [GGM84, GGM86])**.** For functions $t, m, n : \mathbb{N} \to \mathbb{N}$ and $\varepsilon : \mathbb{N} \to [0,1]$, we say that a function $\mathsf{PRF} : \{0,1\}^{m(\lambda)} \times \{0,1\}^{\lambda} \to \{0,1\}^{n(\lambda)}$ is an $(t,\varepsilon)$-secure *pseudorandom function* (PRF) if it holds for every algorithm $\mathcal{D}$ running in time $t(\lambda)$ that

$$\left| \Pr_{\mathsf{K}^{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^{m(\lambda)}} \left[ \mathcal{D}^{\mathsf{PRF}(\mathsf{K}^{\mathsf{PRF}},\cdot)}(1^{\lambda}) = 1 \right] - \Pr_{F \xleftarrow{\$} \mathcal{F}_{\lambda,n(\lambda)}} \left[ \mathcal{D}^{F(\cdot)}(1^{\lambda}) = 1 \right] \right| \leq \varepsilon(\lambda),$$

where $\mathcal{F}_{\lambda,n(\lambda)} = \{F : \{0,1\}^{\lambda} \to \{0,1\}^{n(\lambda)}\}$ is the set of all functions from $\{0,1\}^{\lambda}$ to $\{0,1\}^{n(\lambda)}$.

For a clear exposition of our reduction strategy, assume that all queries by the adversary and the challenge are passed through a truly random function $F \xleftarrow{\$} \mathcal{F}_{\lambda,n(\lambda)}$. We later replace this truly random function with a PRF. We then show that if the PRF is secure, then this does only make a negligible difference in the success probability.

We use the outputs $X'$ of the truly random function for partitioning in the following way. The reduction samples $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \{0,1\}^{\eta}$ for some carefully chosen $\eta \in [n(\lambda)]$. It then defines the uncontrolled set, *i.e.*, the set of inputs for which the reduction can extract a solution but not answer evaluation queries, as the set of all inputs whose PRF output match $\mathsf{K}^{\mathsf{part}}$ on the first $\eta$ bits. We formalize this partitioning as the following function $\mathsf{F}$.

**Definition 29.** For $X' \in \{0,1\}^{n(\lambda)}$ and $\mathsf{K}^{\mathsf{part}} \in \{0,1\}^{\eta}$, we define

$$\mathsf{F}(X', \mathsf{K}^{\mathsf{part}}) := \begin{cases} 1 & \text{if } X'_{|\eta} = \mathsf{K}^{\mathsf{part}} \\ 0 & \text{otherwise,} \end{cases}$$

where $X'_{|\eta}$ denotes the first $\eta$ bits of $X'$.

We have previously used a function $\mathsf{F}$ with the same name in Chapter 3 in the context of admissible hash functions. However, since there is no risk of confusion due to the context and the functions capturing a similar intuition, we reuse the symbol $\mathsf{F}$ here.

Let $\mathsf{TRF} \xleftarrow{\$} \mathcal{F}_{\lambda,n(\lambda)}$ be a truly random function and let $X^{(1)}, \ldots, X^{(Q)}, X^* \in \{0,1\}^\lambda$ be arbitrary with $X^{(i)} \neq X^{(j)}$ and $X^{(i)} \neq X^*$ for all $i \neq j$. We then let $X'_i := \mathsf{TRF}(X^{(i)})$ and $X^{*'} := \mathsf{TRF}(X^*)$. Observe that we then have that all $X'_i$ and $X^{*'}$ are independent and uniformly random in $\{0,1\}^{n(\lambda)}$. We show in the following Lemma that for $\eta = \lceil \log(Q) \rceil + 1$ and $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \{0,1\}^\eta$, where $Q$ is the number of evaluation queries made by the adversary, we have that $\mathsf{F}(X'_i, \mathsf{K}^{\mathsf{part}}) = 0$ for all $i \in [Q]$ and $\mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1$ with probability at least $1/(8Q)$. That means, the partitioning argument has optimal tightness for VRFs up to a small constant factor. We later on show that since a pseudorandom function is indistinguishable from a truly random function, we can efficiently apply this in our construction.

**Lemma 10.** *Let $Q = Q(\lambda)$ be a polynomial, let $\eta = \eta(\lambda) := \lceil \log(Q) \rceil + 1$ and let $X'_1, \ldots, X'_Q, X^{*'}$ be as above. For $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \{0,1\}^\eta$, we then have that*

$$\Pr\left[\mathsf{F}(X'_i, \mathsf{K}^{\mathsf{part}}) = 0 \text{ for all } 0 \leq i \leq Q \text{ and } \mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \geq \frac{1}{8Q}.$$

*Proof.* We start by lower bounding the probability from the lemma as follows.

$$\begin{aligned}
&\Pr\left[\mathsf{F}(X'_i, \mathsf{K}^{\mathsf{part}}) = 0 \text{ for all } 0 \leq i \leq Q \text{ and } \mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \\
&= \Pr\left[\mathsf{F}(X'_i, \mathsf{K}^{\mathsf{part}}) = 0 \text{ for all } 0 \leq i \leq Q \mid \mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \Pr\left[\mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \\
&= \left(\prod_{i=1}^{Q} \Pr\left[\mathsf{F}(X'_i, \mathsf{K}^{\mathsf{part}}) \mid \mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right]\right) \Pr\left[\mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \qquad (4.1) \\
&= \left(1 - \left(\frac{1}{2}\right)^\eta\right)^Q \Pr\left[\mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \\
&\geq \left(1 - \left(\frac{1}{2}\right)^\eta Q\right) \Pr\left[\mathsf{F}_{\mathsf{K}}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 1\right] \qquad (4.2) \\
&= \left(1 - \left(\frac{1}{2}\right)^\eta Q\right) \left(\frac{1}{2}\right)^\eta
\end{aligned}$$

Observe that Equation (4.1) holds because all $X'_i$ and $X^{*'}$ are stochastically independent and that Equation (4.2) follows from Bernoulli's inequality. Next, notice that since $\eta = \lceil \log(Q) \rceil + 1$ we have that $\left(\frac{1}{2}\right)^\eta \geq \left(\frac{1}{2}\right)^{\log(Q)+2} = \frac{1}{4Q}$ and $-\left(\frac{1}{2}\right)^\eta \geq -\left(\frac{1}{2}\right)^{\log(Q)+1} = -\frac{1}{2Q}$. We can therefore conclude the proof as follows.

$$\begin{aligned}
&\Pr\left[\mathsf{F}(X^{(i)}, \mathsf{K}^{\mathsf{part}}) = 0 \text{ for all } 0 \leq i \leq Q \text{ and } \mathsf{F}(X^*, \mathsf{K}^{\mathsf{part}}) = 1\right] \\
&\geq \left(1 - \left(\frac{1}{2}\right)^\eta Q\right) \left(\frac{1}{2}\right)^\eta \geq \left(1 - \frac{1}{2Q} Q\right) \frac{1}{4Q} = \frac{1}{2} \cdot \frac{1}{4Q} = \frac{1}{8Q}
\end{aligned}$$

□

Note that Lemma 10 only holds if all $X'_i$ and $X^{*'}$ are distributed independently and uniformly at random in $\{0,1\}^n$, *e.g.*, if $X'_i = \mathsf{TRF}(X^{(i)})$ for all $i \in [Q]$ and $X^{*'} = \mathsf{TRF}(X^*)$. Observe that we stated our argument for a truly random function instead of a PRF and our construction in Section 4.4.2 uses a PRF. Therefore, we define the function $\mathsf{G}$, which uses a pseudorandom function instead of a truly random function.

**Definition 30.** For $X \in \{0,1\}^\lambda, \mathsf{K}^{\mathsf{PRF}} \in \{0,1\}^m$ and $\mathsf{K}^{\mathsf{part}} \in \{0,1\}^\eta$, we define

$$\mathsf{G}(X, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) := \mathsf{F}(\mathsf{PRF}(\mathsf{K}^{\mathsf{PRF}}, X), \mathsf{K}^{\mathsf{part}}).$$

Intuitively, Lemma 10 also applies to $\mathsf{G}$ and adversarially chosen $X^{(i)}$ and $X^*$ because the outputs of the pseudorandom function are indistinguishable from the outputs of a truly random function. Hence, any adversary that is able to efficiently make queries to the PRF such that the probability in Lemma 10 differs significantly from the probability for a truly random function would also be able to distinguish the pseudorandom function from a truly random function. We show that this also holds formally as part of the security proof of the pseudorandomness of our VRF in Section 4.4.2.

### 4.4.2 Verifiable Random Functions with Optimal Tightness

In order to embed the partitioning argument we described in Section 4.4.1 into a VRF, we use the VRF that Yamada describes in [Yam17b, Appendix C]. This is the full version of [Yam17a]. This VRF is well-suited for our purposes, because it enables us to embed the homomorphic evaluation of arbitrary $\mathsf{NAND}$ circuits in the reduction such that the reduction can answer all queries for inputs on which the circuit evaluates to zero and can extract a solution to the underlying complexity assumption for all inputs for which the circuit evaluates to 1. At the same time, the embedding of the circuit hides some input bits, all internal states and the output of the circuit from the adversary. We use this property to embed the homomorphic evaluation of $\mathsf{G}$ from Definition 30. We first describe how we model NAND circuits and then describe the VRF.

NAND CIRCUITS. Before describing our construction, we require a formal definition of $\mathsf{NAND}$ circuits. The type of circuits we consider take two types of inputs: public inputs and secret inputs. For the function $\mathsf{G}$, which we want to embed in the VRF, we can think of the public input as a VRF input $X \in \{0,1\}^\lambda$ and of the secret input as the PRF key $\mathsf{K}^{\mathsf{PRF}}$ and the partitioning key $\mathsf{K}^{\mathsf{part}}$. Like Yamada, we roughly follow the notation of [BHR12] when describing $\mathsf{NAND}$ circuits. That is, we assign an index to each input bit and to each gate, beginning with the public input bits, continuing with the secret inputs bits and finally indexing the gates. Formally, if there are $k \in \mathbb{N}$ inputs of which $k_{\mathsf{pub}} \in [k]$ are public input bits and $k_{\mathsf{sec}} = k - k_{\mathsf{pub}}$ are secret input bits, then we set $\mathcal{P} := [k_{\mathsf{pub}}]$ and $\mathsf{S} := [k_{\mathsf{pub}} + 1, k_{\mathsf{pub}} + k_{\mathsf{sec}}]$ as the respective index sets for the public and secret input bits.

For a NAND circuit $C : \{0,1\}^{|\mathcal{P}|+|\mathcal{S}|} \to \{0,1\}$ with $c$ many gates and $|\mathcal{P}| + |\mathcal{S}|$ many input bits, we assign an index $j \in \mathcal{C} \coloneqq [|\mathcal{P}| + |\mathcal{S}| + 1, |\mathcal{P}| + |\mathcal{S}| + c]$ to each gate. Further, we formalize the wiring of the circuit with the functions $\mathsf{in}_1, \mathsf{in}_2 : \mathcal{C} \to \mathcal{P} \cup \mathcal{S} \cup \mathcal{C}$ that represent the input wires of a gate. We require that for all $j \in \mathcal{C}$ it holds that $\mathsf{in}_1(j) < j$ and $\mathsf{in}_2(j) < j$. This condition ensures that the circuit does not contain any circles.

Since we only consider circuits with a single output bit, we assume without loss of generality that the output of the gate with index $|\mathcal{P}| + |\mathcal{S}| + |\mathcal{C}|$ outputs the overall output of the circuit. Furthermore, we define the depth of a gate $j$ as the maximal distance from any input gate to $j$. Consequentially, we define the depth of a circuit $C$ as the depth of the gate with index $|\mathcal{P}| + |\mathcal{S}| + |\mathcal{C}|$.

Evaluating a circuit. For a circuit $C$ in the notation above with public inputs $\mathbf{p} = (p_j)_{j\in\mathcal{P}}$, secret inputs $\mathbf{s} = (s_j)_{j\in\mathcal{S}}$, gates with indexes in $\mathcal{C}$ and the wiring encoded by $\mathsf{in}^1, \mathsf{in}^2 : \mathcal{C} \to \mathcal{P}\cup\mathcal{S}\cup\mathcal{C}$, we define the function $\mathsf{value} : \mathcal{P}\cup\mathcal{S}\cup\mathcal{C} \to \{0,1\}$ as follows. For all $j \in \mathcal{P}$ we set $\mathsf{value}(j) \coloneqq p_j$ and for all $j \in \mathcal{S}$ as $\mathsf{value}(j) \coloneqq s_j$. Further, for all $j \in \mathcal{C}$, we set $\mathsf{value}(j) \coloneqq \mathsf{value}(\mathsf{in}^1(j))\,\mathsf{NAND}\,\mathsf{value}(\mathsf{in}^2(j))$. In order to evaluate a circuit on input $\mathbf{p} \in \{0,1\}^{|\mathcal{P}|}$ and $\mathbf{s} \in \{0,1\}^{|\mathcal{S}|}$, we compute $\mathsf{value}(|\mathcal{P}| + |\mathcal{S}| + |\mathcal{C}|)$ since the gate with index $|\mathcal{P}| + |\mathcal{S}| + |\mathcal{C}|$ outputs the overall output of $C$. Note that the evaluation of the circuit is well-defined because we have that for all $j \in \mathcal{C}$ it holds that $\mathsf{in}^1(j) < j$ and $\mathsf{in}^2(j) < j$.

Representing G as a circuit. For our construction, we need to represent G from Definition 30 as a NAND-circuit. However, given the plain definition of G, the number of input bits of the circuit depends on $\eta(\lambda)$, which in turn depends on the number $Q$ of Eval queries made by the adversary. We address this by adapting the encoding of $\mathsf{K}^{\mathsf{part}}$. Namely, we let $\mathsf{PrtSmp}(1^\lambda, Q(\lambda))$ be the algorithm that samples $\mathsf{K}^{\mathsf{match}} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$, computes $\eta \coloneqq \lceil \log(Q(\lambda)) \rceil + 1$ sets $\mathsf{K}^{\mathsf{fixing}} = 1^\eta || 0^{n(\lambda)-\eta(\lambda)}$ and outputs $\mathsf{K}^{\mathsf{part}} = (\mathsf{K}^{\mathsf{match}}, \mathsf{K}^{\mathsf{fixing}}) \in (\{0,1\}^{n(\lambda)})^2$. We then adapt the function $\mathsf{F}(X', \mathsf{K}^{\mathsf{part}})$ to compare $X$ and $\mathsf{K}^{\mathsf{match}}$ on all positions where $\mathsf{K}^{\mathsf{fixing}}$ is 1 and output 1 if they match on all such positions and 0 otherwise. These adaptations do not change the output of F or G but ensure that the NAND-circuit representing G only depends on $\lambda$ and not on $Q$. Note that it would be possible to encode $\mathsf{K}^{\mathsf{fixing}}$ more efficiently, but we use this encoding for clarity.

Construction. We assume that the NAND circuits for the function G for different security parameters are publicly known, and we denote the circuit for G with security parameter $\lambda$ by $C_{\mathsf{G},\lambda}$. For our construction, we have that $\mathcal{P} = [\lambda]$, since the public input of G is $X \in \{0,1\}^\lambda$. Furthermore, we set $\mathcal{S}^{\mathsf{PRF}} \coloneqq [|\mathcal{P}| + 1, |\mathcal{P}| + m(\lambda)]$ for the indexes of the bits of $\mathsf{K}^{\mathsf{PRF}} \in \{0,1\}^{m(\lambda)}$, $\mathcal{S}^{\mathsf{part}} \coloneqq [|\mathcal{P}| + |\mathcal{S}^{\mathsf{PRF}}| + 1, |\mathcal{P}| + |\mathcal{S}^{\mathsf{PRF}}| + 2n(\lambda)]$ for the indexes of $\mathsf{K}^{\mathsf{part}} \in \{0,1\}^{2n(\lambda)}$, and $\mathcal{S} \coloneqq \mathcal{S}^{\mathsf{PRF}} \cup \mathcal{S}^{\mathsf{part}}$. Finally, we assume that the functions $\mathsf{in}^1_\lambda, \mathsf{in}^2_\lambda : \mathcal{C} \to \mathcal{P} \cup \mathcal{S} \cup \mathcal{C}$ encode the wiring of $C_{\mathsf{G},\lambda}$

and that $|\mathcal{P}| + |\mathcal{S}| + |\mathcal{C}|$ is the index of the output gate. For simplicity, we set $\mathsf{out} \coloneqq |\mathcal{P}| + |\mathcal{S}| + |\mathcal{C}|$.

**Construction 3.** Let $\mathsf{GrpGen}$ be a certified bilinear group generator (see Definitions 9 and 10) with verification algorithms $\mathsf{GrpVfy}$ and $\mathsf{GrpElemVfy}$. We then let $\mathcal{VRF}_{\mathsf{opt}} = (\mathsf{SetupVRF}_{\mathsf{opt}}, \mathsf{Eval}_{\mathsf{opt}}, \mathsf{Vfy}_{\mathsf{opt}})$ be the following verifiable random function.

$\mathsf{SetupVRF}_{\mathsf{opt}}(1^\lambda)$ first generates a group description $\mathcal{BG} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$ and samples uniformly random group generators $g, h \xleftarrow{\$} \mathbb{G}^*$, $w_0 \xleftarrow{\$} \mathbb{Z}_p^*$ and $w_j \xleftarrow{\$} \mathbb{Z}_p$ for all $j \in \mathcal{S}$. It then sets $W_0 \coloneqq g^{w_0}$, $W_j \coloneqq g^{w_j}$ for all $j \in \mathcal{S}$ and outputs

$$\mathsf{vk} \coloneqq \left(\mathcal{BG}, g, h, W_0, (W_j)_{j\in\mathcal{S}}\right) \qquad \text{and} \qquad \mathsf{sk} \coloneqq \left(w_0, (w_j)_{j\in\mathcal{S}}\right).$$

$\mathsf{Eval}_{\mathsf{opt}}(\mathsf{sk}, X)$ parses $X \in \{0,1\}^\lambda$ as $(X_1, \ldots, X_\lambda)$ and sets

$$\theta_j \coloneqq \begin{cases} X_j & \text{if } j \in \mathcal{P} \\ w_j & \text{if } j \in \mathcal{S} \end{cases}$$

for all $j \in \mathcal{P} \cup \mathcal{S}$. For all $j \in \mathcal{C}$, it sets

$$\theta_j \coloneqq 1 - \theta_{\mathsf{in}^1_\lambda(j)}\theta_{\mathsf{in}^2_\lambda(j)}.$$

It then sets $\pi_0 \coloneqq g^{\theta_{\mathsf{out}}/w_0}$ and $\pi_j \coloneqq g^{\theta_j}$ for all $j \in \mathcal{C}$ and outputs

$$Y \coloneqq e(g,h)^{\theta_{\mathsf{out}}/w_0} \qquad \text{and} \qquad \pi \coloneqq (\pi_0, (\pi_j)_{j\in\mathcal{C}}).$$

$\mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y, \pi)$ verifies that $\mathsf{vk}$ has the form $(\mathcal{BG}, g, h, W_0, (W_j)_{j\in\mathcal{S}})$ and that $\pi$ has the form $(\pi_0, (\pi_j)_{j\in\mathcal{C}})$. It then verifies the group description by running $\mathsf{GrpVfy}(1^\lambda, \mathcal{BG})$ and then verifies all group elements in $\mathsf{vk}$, $\pi$ and $Y$ by running $\mathsf{GrpElemVfy}(1^\lambda, \mathcal{BG}, s)$ for all $s \in \{g, h, Y, \pi_0, \pi_{|\mathcal{P}|+|\mathcal{S}|+1}, \ldots, \pi_{|\mathcal{P}|+|\mathcal{S}|+|\mathcal{C}|}\}$. $\mathsf{Vfy}_{\mathsf{opt}}$ outputs 0 if any of the checks fails. Next, the algorithm verifies the correctness of $Y$ in respect to $\mathsf{vk}$, $X$ and $\pi$ by setting $\pi_j \coloneqq g^{X_j}$ for all $j \in \mathcal{P}$ and $\pi_j \coloneqq W_j$ for all $i \in \mathcal{S}$ and performing the following steps.

1. It checks whether $e(g, \pi_j) = e(g,g)\left(e(\pi_{\mathsf{in}^1_\lambda(j)}, \pi_{\mathsf{in}^2_\lambda(j)})\right)^{-1}$ for all $j \in \mathcal{C}$.
2. It checks whether $e(\pi_0, W_0) = e(\pi_{\mathsf{out}}, g)$.
3. It checks whether $e(\pi_0, h) = Y$.

If any of the checks above fail, then $\mathsf{Vfy}_{\mathsf{opt}}$ outputs 0. Otherwise, it outputs 1.

Instantiation. In order to instantiate $\mathcal{VRF}_{\mathsf{opt}}$, we need that $\mathsf{G}$ can be represented by a circuit of polynomial size and logarithmic depth. While this is certainly possible for the comparison of the PRF output with $\mathsf{K}^{\mathsf{match}}$, we also require a PRF that can be computed by such a NAND circuit. The Naor-Reingold PRF is an example of such a PRF that is also provably secure under the DDH assumption [NR97]. However, we can further optimize the efficiency by using the adaptation of the Naor-Reingold PRF in [JKP18, Section 5.1]. This PRF has secret keys of size $\omega(\log(\lambda))$. Further, we can change the encoding of $\mathsf{K}^{\mathsf{match}}$ and $\mathsf{K}^{\mathsf{fixing}}$ to also consist of only $\omega(\log(\lambda))$ many bits. This would bring the size of the public verification key down to $\omega(\log(\lambda))$, would, however, only hold for $\lambda$ large enough. We can further optimize the size of the proofs by applying the technique of [IKOS08], which allows to reduce the circuit size of every PRF to $\mathcal{O}(\lambda)$ at the cost of reducing the output length to $\lambda^{1/c}$ for some constant $c > 0$ that depends on the PRF. However, the smaller output length is no issue, since $\lambda^{1/c}$ is larger than $\lceil \log(Q(\lambda)) \rceil + 1 = \mathcal{O}(\log(\lambda))$ for large enough $\lambda$, because $Q$ is polynomial in $\lambda$. This technique therefore reduces the size of proofs to $\mathcal{O}(\lambda)$. The author wants to thank Yuval Ishai for the helpful discussion on optimizing the size of the circuit.

**Correctness and Unique Provability of the VRF.**

The proofs for correctness and unique provability closely follow the respective proofs by Yamada [Yam17b, Appendix C]. Therefore, we only present them here for completeness. Before proving the pseudorandomness of the VRF, we shortly discuss the instantiation with concrete PRFs and the effect on the efficiency.

Correctness. We prove the correctness of $\mathcal{VRF}_{\mathsf{opt}}$ by considering an arbitrary input $X \in \{0,1\}^\lambda$. Let $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}_{\mathsf{opt}}(1^\lambda)$ and $(Y, \pi) \coloneqq \mathsf{Eval}_{\mathsf{opt}}(\mathsf{sk}, X)$, then the algorithm $\mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y, \pi)$ first verifies the structure of $\mathsf{vk}$ and $\pi$. This verification succeeds because $\mathsf{vk}$ and $\pi$ are generated in this specific format by $\mathsf{SetupVRF}_{\mathsf{opt}}$ and $\mathsf{Eval}_{\mathsf{opt}}$. The same applies to the verification of $\mathcal{BG}$ and the encoding of the group elements by $\mathsf{GrpVfy}$ and $\mathsf{GrpElemVfy}$. Further, the first check succeeds because $\mathsf{Eval}_{\mathsf{opt}}$ computes $\pi_j$ for all $j \in \mathcal{C}$ such that

$$\begin{aligned} e(g, \pi_j) = e(g, g^{\theta_j}) = e(g, g^{1-\theta_{\mathsf{in}^1_\lambda(j)}\theta_{\mathsf{in}^2_\lambda(j)}}) = e(g,g)e\left(g, g^{\theta_{\mathsf{in}^1_\lambda(j)}\theta_{\mathsf{in}^2_\lambda(j)}}\right)^{-1} \\ = e(g,g)e\left(g^{\theta_{\mathsf{in}^1_\lambda(j)}}, g^{\theta_{\mathsf{in}^2_\lambda(j)}}\right)^{-1} = e(g,g)e\left(\pi_{\mathsf{in}^1_\lambda(j)}, \pi_{\mathsf{in}^2_\lambda(j)}\right)^{-1}. \end{aligned}$$

Further, the second check succeeds because $\mathsf{Eval}_{\mathsf{opt}}$ and $\mathsf{SetupVRF}_{\mathsf{opt}}$ compute $\pi_0, \pi_{\mathsf{out}}$ and $W_0$ such that $e(\pi_0, W_0) = e(g^{\theta_{\mathsf{out}}/w_0}, g^{w_0}) = e(g^{\theta_{\mathsf{out}}}, g) = e(\pi_{\mathsf{out}}, g)$. Finally, we have that

$$e(\pi_0, h) = e(g^{\theta_{\mathsf{out}}/w_0}, h) = e(g,h)^{\theta_{\mathsf{out}}/w_0} = Y.$$

Therefore, $\mathsf{Vfy}_{\mathsf{opt}}$ outputs 1, which proves the correctness of $\mathcal{VRF}_{\mathsf{opt}}$.

Unique Provability. In order to show that $\mathcal{VRF}_{\mathsf{opt}}$ has unique provability, we have to show that for every $\mathsf{vk} \in \{0,1\}^*$ and $X \in \{0,1\}^\lambda$ there does not exist $Y^0, \pi^0, Y^1, \pi^1 \in \{0,1\}^*$ with $Y^0 \neq Y^1$ such that $\mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y^0, \pi^0) = \mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y^1, \pi^1) = 1$.

We do so by assuming that there are $\mathsf{vk}, Y^0, \pi^0, Y^1, \pi^1 \in \{0,1\}^*$ such that $\mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y^0, \pi^0) = \mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y^1, \pi^1) = 1$ and then conclude that $Y^0 = Y^1$ has to hold by going through the checks of the verification algorithm. $\mathsf{Vfy}_{\mathsf{opt}}$ first checks whether $\mathsf{vk}$ and $\pi$ both have the correct format and that supposed group elements in $\mathsf{vk}, \pi$ and $Y$ are actual group elements with a unique encoding. Since we assumed that $\mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y^0, \pi^0) = \mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y^1, \pi^1) = 1$, we from now on assume that $\mathsf{vk}, Y^0, \pi^0, Y^1, \pi^1$ fulfill these conditions.

Next, observe that it follows from the group structure of $\mathbb{G}$ that $\pi_j = g^{X_j}$ is uniquely defined for all $j \in \mathcal{P}$ and that $\log_g(\pi_j) = w_j$ uniquely defines $w_j \in \mathbb{Z}_p$ for all $j \in \mathcal{S}$. $\mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y_0, \pi_0) = \mathsf{Vfy}_{\mathsf{opt}}(\mathsf{vk}, X, Y_1, \pi_1) = 1$. Then, the first check of $\mathsf{Vfy}_{\mathsf{opt}}$ inductively specifies a unique $\pi_j \in \mathbb{G}$ such that $e(g, \pi_j) = e(g,g)\left(e(\pi_{\mathsf{in}^1_\lambda(j)}, \pi_{\mathsf{in}^2_\lambda(j)})\right)^{-1}$ holds for all $j \in \mathcal{C}$. This implies that the values $\pi_j$ are identical in $\pi^0$ and $\pi^1$. The second check of $\mathsf{Vfy}_{\mathsf{opt}}$ then uniquely specifies $\pi_0$, because $W_0$ and $\pi_{\mathsf{out}}$ are uniquely specified. Hence, $\pi_0$ has to be identical in $\pi^0$ and $\pi^1$. Finally, the last check of $\mathsf{Vfy}_{\mathsf{opt}}$ uniquely specifies $Y$ because $\pi_0$ is already unique. Therefore, $Y^0 = Y^1$ has to hold, which proves the unique provability of $\mathcal{VRF}_{\mathsf{opt}}$.

**Proof of Pseudorandomness.**

The security of our VRF is based on the decisional $q$-bilinear Diffie-Hellman inversion assumption that we introduced in Section 2.4 as Definition 14. Note that the assumption falls in the category of non-interactive complexity assumptions from Definition 24. Based on this assumption, we can formulate the theorem for the pseudorandomness of our VRF.

**Theorem 8.** *Let $\mathcal{VRF}_{\mathsf{opt}} = (\mathsf{SetupVRF}_{\mathsf{opt}}, \mathsf{Eval}_{\mathsf{opt}}, \mathsf{Vfy}_{\mathsf{opt}})$ be the verifiable random function from Construction 3, then for every legitimate adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ breaks the pseudorandomness of $\mathcal{VRF}_{\mathsf{opt}}$ and makes $Q(\lambda)$ queries to $\mathsf{Eval}_{\mathsf{opt}}$ for some polynomial $Q : \mathbb{N} \to \mathbb{N}$, there exists an algorithm $\mathcal{B}$ that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}})$-breaks the $q$-DBDHI assumption relative to $\mathsf{GrpGen}$ used in $\mathcal{VRF}_{\mathsf{opt}}$ with*

$$t_{\mathcal{B}}(\lambda) = t_{\mathcal{A}}(\lambda), \qquad \varepsilon_{\mathcal{B}}(\lambda) \geq \frac{\varepsilon_{\mathcal{A}}(\lambda)}{8Q(\lambda)} - \varepsilon_{\mathsf{PRF}}(\lambda) - \mathsf{negl}(\lambda) \qquad \text{and} \qquad q \coloneqq 2^d,$$

*where $d$ is the depth of $C_{\mathsf{G},\lambda}$, $\varepsilon_{\mathsf{PRF}}$ is the largest advantage any algorithm with runtime $t_{\mathcal{A}}(\lambda)$ that makes $Q(\lambda)$ queries to its oracle has in breaking the security of the $\mathsf{PRF}$ used in $\mathcal{VRF}_{\mathsf{opt}}$ and $\mathsf{negl}(\lambda)$ is a negligible function. In particular: $\mathcal{VRF}_{\mathsf{opt}}$ achieves the optimal tightness, since $\varepsilon_{\mathsf{PRF}}(\lambda)$ is negligible if the construction is instantiated with a PRF with a security reduction loss of at most $Q(\lambda)$.*

*Remark* 9. Note that the requirement of a loss of at most $Q$ for the PRF is fulfilled by *e.g.* the Naor-Reingold PRF [NR97] or the PRFs by Jager *et al.* [JKP18].

*Proof.* Since $\mathsf{Eval}_{\mathsf{opt}}$ is deterministic, $\mathcal{A}$ can not learn anything by making the same query to $\mathsf{Eval}_{\mathsf{opt}}$ twice. We therefor assume without loss of generality that $\mathcal{A}$ makes only pairwise distinct queries to $\mathsf{Eval}_{\mathsf{opt}}$. Further, we set $Q := Q(\lambda), n := n(\lambda), m := m(\lambda)$ and $\varepsilon_{\mathcal{A}} := \varepsilon_{\mathcal{A}}(\lambda)$ in order to simplify notation.

We denote the event that Game $i$ outputs 1 by $\mathsf{G}_i$. The first part of the proof will focus on our technique of using a PRF for partitioning. The second part of the proof follows the proof by Yamada [Yam17b, Theorem 6] and we provide it for completeness.

*Game 0.* This is the original security experiment from Definition 6 and we thus have that

$$\left|\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right| = \varepsilon_{\mathcal{A}}$$

holds by definition.

*Game 1.* In this game, the challenger first runs the game as before. But, before outputting a result, it samples $X_i' \xleftarrow{\$} \{0,1\}^n$ uniformly and independently at random for each query $X^{(i)} \in \{0,1\}^\lambda$ to $\mathsf{Eval}_{\mathsf{opt}}$ by $\mathcal{A}$ and $X^{*'} \xleftarrow{\$} \{0,1\}^n$ for the challenge $X^* \in \{0,1\}^\lambda$. Observe that this perfectly emulates the process of evaluating a truly random function on the queries and the challenge because we assumed without loss generality that all queries and the challenge are pairwise distinct. Further, it sets $\eta := \lceil \log Q \rceil + 1$ and samples $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q)$. It then aborts and outputs a random bit if $\mathsf{F}(X_i', \mathsf{K}^{\mathsf{part}}) = 1$ for any $i \in [Q]$ or if $\mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 0$. We denote the occurrence of any of the two abort conditions by the event $\mathsf{bad}$. We next show that

$$\left|\Pr\left[\mathsf{G}_1\right] - \Pr\left[\mathsf{G}_0\right]\right| = \varepsilon_{\mathcal{A}}(1 - \Pr\left[\mathsf{bad}\right]) \leq \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right)$$

holds. We later use that $\Pr\left[\neg\mathsf{bad}\right] \geq 1/(8Q)$ holds, which follows from Lemma 10 and will in the end yield the loss stated in Theorem 8. We have the following.

$$\begin{aligned}
\left|\Pr\left[\mathsf{G}_1\right] - \Pr\left[\mathsf{G}_0\right]\right| &= \left|\Pr\left[\mathsf{G}_1 \mid \mathsf{bad}\right]\Pr\left[\mathsf{bad}\right] + \Pr\left[\mathsf{G}_1 \mid \neg\mathsf{bad}\right]\Pr\left[\neg\mathsf{bad}\right] - \Pr\left[\mathsf{G}_0\right]\right| \\
&= \left|\frac{1}{2}\left(1 - \Pr\left[\neg\mathsf{bad}\right]\right) + \Pr\left[\mathsf{G}_1 \mid \neg\mathsf{bad}\right]\Pr\left[\neg\mathsf{bad}\right] - \Pr\left[\mathsf{G}_0\right]\right| \\
&= \left|\frac{1}{2} + \Pr\left[\neg\mathsf{bad}\right]\left(\Pr\left[\mathsf{G}_1 \mid \neg\mathsf{bad}\right] - \frac{1}{2}\right) - \Pr\left[\mathsf{G}_0\right]\right| \\
&= \left|\frac{1}{2} + \Pr\left[\neg\mathsf{bad}\right]\left(\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right) - \Pr\left[\mathsf{G}_0\right]\right| \qquad (4.3) \\
&= \left|\Pr\left[\neg\mathsf{bad}\right]\left(\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right) - \left(\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right)\right| \\
&= \left|\left(\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right)\left(\Pr\left[\neg\mathsf{bad}\right] - 1\right)\right| \\
&= \left|\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right| \cdot \left|\Pr\left[\neg\mathsf{bad}\right] - 1\right| \\
&= \varepsilon_{\mathcal{A}} \cdot \left(1 - \Pr\left[\neg\mathsf{bad}\right]\right)
\end{aligned}$$

Note that Equation (4.3) holds because $\Pr[\mathsf{G}_1 \mid \neg\mathsf{bad}] = \Pr[\mathsf{G}_0 \mid \neg\mathsf{bad}]$ and the event $\neg\mathsf{bad}$ is independent of $\mathsf{G}_0$. The independence holds because $X^{*'}$ and all $X_i'$ are drawn at random. Note that it is this independence together with the independence between the different $X_i'$ and $X^*$ that allows us to achieve the optimal tightness in contrast to the other approaches discussed in the introduction.

Further, by Lemma 10, we have that $\Pr[\neg\mathsf{bad}] \geq 1/(8Q)$ holds and therefore holds that

$$|\mathsf{G}_1 - \mathsf{G}_0| = \varepsilon_{\mathcal{A}}(1 - \Pr[\neg\mathsf{bad}]) \leq \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right).$$

*Game 2.* In this game, the challenger only changes the way it computes $X^{*'}$ and $X_i'$ for all $i \in [Q]$. The challenger samples $\mathsf{K}^{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^m$ and aborts and outputs a random bit if $\mathsf{G}(X^{(i)}, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 1$ or if $\mathsf{G}(X^*, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 0$. The only difference to Game 1 is that $\mathsf{G}$ sets $X^{*'} \coloneqq \mathsf{PRF}(\mathsf{K}^{\mathsf{PRF}}, X^*)$ and $X_i' \coloneqq \mathsf{PRF}(\mathsf{K}^{\mathsf{PRF}}, X^{(i)})$ instead of drawing them uniformly at random.

Informally, every algorithm distinguishing Game 2 from Game 1 with advantage $\varepsilon$ implies a distinguisher for $\mathsf{PRF}$ with advantage $\varepsilon$. We describe a distinguisher $\mathcal{B}_{\mathsf{PRF}}$ for $\mathsf{PRF}$ that is based on Game 2 and Game 1 and achieves exactly this: $\mathcal{B}_{\mathsf{PRF}}(\lambda)$ with access to either a $\mathsf{PRF}(\mathsf{K}^{\mathsf{PRF}}, \cdot)$ or a truly random function $F \xleftarrow{\$} \mathcal{F}_{\lambda,n(\lambda)}$ as oracle first runs $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupVRF}_{\mathsf{opt}}(1^\lambda)$ and uses $\mathsf{sk}$ to answer all queries and the challenge by $\mathcal{A}$. After $\mathcal{A}$ submits its guess $b'$, $\mathcal{B}_{\mathsf{PRF}}$ queries its oracle on $X^{(i)}$ and by that obtains $X_i'$ for all $i \in [Q]$. Analogously, it queries its oracle on $X^*$ and by that obtains $X^{*'}$. It then samples $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q)$ and aborts and outputs a random bit if $\mathsf{F}(X^{*'}, \mathsf{K}^{\mathsf{part}}) = 0$ or $\mathsf{F}(X_i', \mathsf{K}^{\mathsf{part}}) = 1$ for some $i \in [Q]$. Otherwise, $\mathcal{B}_{\mathsf{PRF}}$ outputs 1 if $\mathcal{A}$'s guess is correct and 0 otherwise.

Note that $\mathcal{B}$ has exactly the same runtime as $\mathcal{A}$ and that the probability that it outputs 1 is identical to $\Pr[\mathsf{G}_2]$ if its oracle is the pseudorandom function. Analogously, if its oracle is a truly random function, then its output is 1 with probability $\Pr[\mathsf{G}_1]$ and therefore

$$|\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_1]| = \left|\Pr_{\mathsf{K}^{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^m}\left[\mathcal{B}_{\mathsf{PRF}}^{\mathsf{PRF}(\mathsf{K}^{\mathsf{PRF}}, \cdot)}(1^\lambda) = 1\right] - \Pr_{F \xleftarrow{\$} \mathcal{F}_{\lambda,n(\lambda)}}\left[\mathcal{B}_{\mathsf{PRF}}^{F(\cdot)} = 1\right]\right| \leq \varepsilon_{\mathsf{PRF}}.$$

*Game 3.* In this game, the challenger samples the key of the PRF $\mathsf{K}^{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^m$ and $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q)$ in the very beginning and aborts and outputs a random bit as soon as $\mathcal{A}$ makes an $\mathsf{Eval}_{\mathsf{opt}}$ query $X^{(i)}$ with $\mathsf{G}(X^{(i)}, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 1$ or if it holds for $\mathcal{A}$'s challenge $X^*$ that $\mathsf{G}(X^*, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 0$. Since this is just a conceptual change, we have that

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

From here on, the proof mostly follows the proof by Yamada [Yam17b, Appendix C] and we present it here for completeness.

*Game 4.* In this game, we change the way the values $w_j$ are chosen. That is, the challenger samples the partitioning key $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q)$ with $\mathsf{K}^{\mathsf{part}} \in \{0,1\}^{|\mathcal{S}^{\mathsf{part}}|}$ and $\mathsf{K}^{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^{|\mathcal{S}^{\mathsf{PRF}}|}$. For all $j \in \mathcal{S}$ it sets $s_j \coloneqq \mathsf{K}^{\mathsf{PRF}}_{j-|\mathcal{P}|}$ for all $j \in \mathcal{S}^{\mathsf{PRF}}$ and $s_j \coloneqq \mathsf{K}^{\mathsf{part}}_{j-|\mathcal{P}|-|\mathcal{S}^{\mathsf{PRF}}|}$ for all $j \in \mathcal{S}^{\mathsf{part}}$. The challenger then samples $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, and $\tilde{w}_j \xleftarrow{\$} \mathbb{Z}_p^*$ for all $j \in \mathcal{S}$. It then sets

$$w_0 \coloneqq \tilde{w}_0 \alpha \qquad \text{and} \qquad w_j \coloneqq \tilde{w}_j \cdot \alpha + s_j \qquad \text{for all } j \in \mathcal{S}.$$

Note that the values $\tilde{w}_j$ are drawn from $\mathbb{Z}_p^*$ and not from $\mathbb{Z}_p$ like the values $w_j$ in the previous game. This slightly changes the distributions of each $w_j$. However, the overall statistical distance is at most $|\mathcal{S}|/p$, which is negligible because $p = \Omega(2^\lambda)$ by Definition 9 and therefore

$$|\mathsf{G}_4 - \mathsf{G}_3| = \mathsf{negl}(\lambda).$$

Helping Definitions. Before proceeding to the next game, we introduce additional notation. That is, for all $X \in \{0,1\}^\lambda$ and all $j \in \mathcal{P} \cup \mathcal{S} \cup \mathcal{C}$, we let

$$\mathsf{P}_{X,j}(\mathsf{Z}) \coloneqq \begin{cases} X_j & \text{if } j \in \mathcal{P}, \\ \tilde{w}_i \mathsf{Z} + s_j & \text{if } j \in \mathcal{S} \text{ and} \\ 1 - \mathsf{P}_{X,\mathsf{in}^1_\lambda(j)}(\mathsf{Z}) \mathsf{P}_{X,\mathsf{in}^2_\lambda(j)}(\mathsf{Z}) & \text{if } j \in \mathcal{C}. \end{cases}$$

Note that by the definition of $w_j$ form Game 3, we have that $\mathsf{P}_{X,j}(\alpha) = \theta_j$. In order to proceed to the next game, we require the following lemma by Yamada.

**Lemma 11** (Lemma 16 in [Yam17b])**.** *There exists* $\mathsf{R}_X(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ *with* $\mathsf{deg}(\mathsf{R}(\mathsf{Z})) \leq \mathsf{deg}(\mathsf{P}_{X,\mathsf{out}}(\mathsf{Z})) \leq 2^d$*, where* $d$ *is the depth of the circuit for the function* $\mathsf{G}$*, and it holds that*

$$\mathsf{P}_{X,\mathsf{out}}(\mathsf{Z}) = \mathsf{G}(X, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) + \mathsf{Z} \cdot \mathsf{R}_X(\mathsf{Z}).$$

As not to interrupt the proof of Theorem 8, we postpone the proof of Lemma 11 and first conclude the proof of Theorem 8.

*Game 5.* With Lemma 11 at our hands, we change how the challenger answers $\mathcal{A}$'s evaluation queries in this game. As in the previous game, the challenger aborts and outputs a random bit if $\mathsf{G}(X^{(i)}, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 1$ for any query $X^{(i)}$ by $\mathcal{A}$. Otherwise, the challenger computes and outputs

$$Y \coloneqq e\left(g^{\mathsf{R}_X(\alpha)/\tilde{w}_0}, h\right), \qquad \pi \coloneqq \left(\pi_0 = g^{\mathsf{R}_X(\alpha)/\tilde{w}_0}, \left(\pi_j \coloneqq g^{\mathsf{P}_{X,j}(\alpha)}\right)_{j \in \mathcal{C}}\right).$$

Observe that $Y$ and $\pi$ are distributed exactly as in Game 4. This holds for all $\pi_j$ because $\mathsf{P}_{X,j}(\mathsf{Z})$ is defined exactly as $\mathsf{P}_j$ in the definition of $\mathsf{Eval}_{\mathsf{opt}}$ above, just with $w_j$ defined as in Game 4. Further, it holds for $\pi_0$ and $Y$ because

$$\frac{\mathsf{R}_X(\alpha)}{\tilde{w}_0} = \frac{\alpha \cdot \mathsf{R}_X(\alpha)}{\alpha \cdot \tilde{w}_0} = \frac{\mathsf{G}(X, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) + \alpha \cdot \mathsf{R}_X(\alpha)}{\alpha \cdot \tilde{w}_0} = \frac{\mathsf{P}_{X,\mathsf{out}}(\alpha)}{w_0},$$

where the last equality follows from Lemma 11. It therefore holds that

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

*Game 6.* In this game, we change how the challenger answers to $\mathcal{A}$'s challenge $X^*$. As in the previous game, the challenger aborts and outputs a random bit if $\mathsf{G}(X^*, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 0$. Otherwise, the challenger computes $\mathsf{R}_{X^*}(\alpha)$ and sets

$$\begin{aligned} Y_0 &:= \left(e(g,h)^{1/\alpha} \cdot e\left(g^{\mathsf{R}_{X^*}(\alpha)}, h\right)\right)^{1/\tilde{w}_0} = e\left(g^{(1+\alpha \mathsf{R}^*_X(\alpha))/(\tilde{w}_0 \alpha)}, h\right) \\ &= e\left(g^{(\mathsf{G}(X^*, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) + \alpha \mathsf{R}^*_X(\alpha))/(\tilde{w}_0 \alpha)}, h\right) = e\left(g^{\mathsf{P}_{X^*,\mathsf{out}}(\alpha)/w_0}, h\right). \end{aligned}$$

Then, the challenger samples a bit $b$ uniformly at random and $Y_1 \xleftarrow{\$} \mathbb{G}_T$ and outputs $Y_b$ to $\mathcal{A}$. Again, observe that $\mathsf{P}_{X^*,\mathsf{out}}(\alpha)$ is, relative to $w_j$ as defined in Game 4, distributed exactly as $\theta_{\mathsf{out}}$ in the definition of $\mathsf{Eval}_{\mathsf{opt}}$. It therefore holds that

$$\Pr[\mathsf{G}_6] = \Pr[\mathsf{G}_5].$$

*Game 7.* In this game, the challenger always responds to $\mathcal{A}$'s challenge $X^*$ with $Y_1 \xleftarrow{\$} \mathbb{G}_T$ regardless of the value of $b$. It thus holds that

$$\Pr[\mathsf{G}_7] = \frac{1}{2},$$

because no response of the challenger to the challenge $X^*$ or an evaluation query $X^{(i)}$ depends in $b$ in any way. Furthermore, we claim that there is an algorithm $\mathcal{B}$ that runs in time $t_{\mathcal{A}}$ and has an advantage of $|\Pr[\mathsf{G}_6] - \Pr[\mathsf{G}_7]|$ in solving the $q$-DBDHI problem.

**Lemma 12.** *Let $d \in \mathbb{N}$ be the depth of the $C_{\mathsf{G},\lambda}$, then there is an algorithm $\mathcal{B}$ with run time $t_{\mathcal{B}} \approx t_{\mathcal{A}}$ that on input a $q$-DBDHI instance with $q = 2^d$ perfectly simulates either Game 6 or Game 7 such that*

$$\varepsilon_{\mathcal{B}} := \mathsf{Adv}^{q\text{-DBDHI}}_{\mathcal{B}}(\lambda) = |\Pr[\mathsf{G}_6] - \Pr[\mathsf{G}_7]|$$

*holds.*

As not to interrupt the proof of Theorem 8, we postpone the proof of Lemma 12 and first conclude the proof of Theorem 8. By Lemma 12 and the (in)equalities we derived above, it holds that

$$\begin{aligned}
\varepsilon_{\mathcal{A}} &= \left|\Pr\left[\mathsf{G}_0\right] - \frac{1}{2}\right| \leq \left|\Pr\left[\mathsf{G}_0\right] - \Pr\left[\mathsf{G}_1\right]\right| + \left|\Pr\left[\mathsf{G}_1\right] - \frac{1}{2}\right| \\
&\leq \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \left|\Pr\left[\mathsf{G}_1\right] - \frac{1}{2}\right| \\
&\leq \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \varepsilon_{\mathsf{PRF}} + \left|\Pr\left[\mathsf{G}_2\right] - \frac{1}{2}\right| \\
&= \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \varepsilon_{\mathsf{PRF}} + \left|\Pr\left[\mathsf{G}_3\right] - \frac{1}{2}\right| \\
&\leq \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \varepsilon_{\mathsf{PRF}} + \mathsf{negl}(\lambda) + \left|\Pr\left[\mathsf{G}_4\right] - \frac{1}{2}\right| \\
&= \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \varepsilon_{\mathsf{PRF}} + \mathsf{negl}(\lambda) + \left|\Pr\left[\mathsf{G}_6\right] - \frac{1}{2}\right| \\
&\leq \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \varepsilon_{\mathsf{PRF}} + \mathsf{negl}(\lambda) + \left|\Pr\left[\mathsf{G}_6\right] - \Pr\left[\mathsf{G}_7\right]\right| + \left|\Pr\left[\mathsf{G}_7\right] - \frac{1}{2}\right| \\
&= \varepsilon_{\mathcal{A}}\left(1 - \frac{1}{8Q}\right) + \varepsilon_{\mathsf{PRF}} + \mathsf{negl}(\lambda) + \varepsilon_{\mathcal{B}}.
\end{aligned}$$

Rearranging the terms above, we have that

$$\varepsilon_{\mathcal{B}} \geq \frac{\varepsilon_{\mathcal{A}}}{8Q} - \varepsilon_{\mathsf{PRF}} - \mathsf{negl}(\lambda),$$

which concludes the proof of Theorem 8. □

**Proofs for Lemma 11 and Lemma 12.**

In order to finalize the proof of Theorem 8, we still need to prove Lemma 11 and Lemma 12, which we do below.

*Proof of Lemma 11.* Our proof closely follows Yamada's proof of Lemma 16 in Appendix C of [Yam17b]. For all $j \in \mathcal{P} \cup \mathcal{S} \cup \mathcal{C}$ let

$$b_j := \begin{cases} X_j & \text{if } j \in \mathcal{P} \\ \mathsf{K}^{\mathsf{PRF}}_{j-|\mathcal{P}|} & \text{if } j \in \mathcal{S}^{\mathsf{PRF}} \\ \mathsf{K}^{\mathsf{part}}_{j-|\mathcal{P}|-|\mathcal{S}^{\mathsf{PRF}}|} & \text{if } j \in \mathcal{S}^{\mathsf{part}} \\ 1 - b_{\mathsf{in}^1_\lambda(j)} \cdot b_{\mathsf{in}^2_\lambda(j)} & \text{if } j \in \mathcal{C}. \end{cases}$$

Note that for two bits $a, b \in \{0,1\}$, it holds that $1 - ab = a \,\mathsf{NAND}\, b$. Therefore, we have for all $j \in \mathcal{C}$ that $b_j$ is the output of gate $j$ and in particular, that $b_{\mathsf{out}} =$

$\mathsf{G}(X, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}})$. We now claim that for all $j \in \mathcal{C}$ there exist $\mathsf{R}_{X,j}(\mathsf{Z}) \in \mathbb{Z}_p[\mathsf{Z}]$ such that

$$\mathsf{P}_{X,j}(\mathsf{Z}) = b_j + \mathsf{Z} \cdot \mathsf{R}_{X,j}(\mathsf{Z}).$$

Furthermore, it holds for all $j \in \mathcal{P} \cup \mathcal{S} \cup \mathcal{C}$, that if $d_j \in \mathbb{N}$ is the depth of $j$, then $\mathsf{deg}(\mathsf{P}_{X,j}(\mathsf{Z})) \leq 2^{d_j}$. We prove this by induction. For all $j \in \mathcal{P} \cup \mathcal{S}$ this holds by the definition of $\mathsf{P}_{X,j}(\mathsf{Z})$. For all $j \in \mathcal{C}$, let $j_1 := \mathsf{in}^1_\lambda(j)$ and $j_2 := \mathsf{in}^2_\lambda(j)$. Note that by our requirements for $\mathsf{in}^1_\lambda$ and $\mathsf{in}^2_\lambda$ we have that $j_1 < j$ and $j_2 < j$, which allows us to prove the statement by induction. We then prove our claim as follows.

$$\begin{aligned}
\mathsf{P}_{X,j}(\mathsf{Z}) &= 1 - \mathsf{P}_{X,j_1}(\mathsf{Z})\mathsf{P}_{X,j_2}(\mathsf{Z}) \\
&= 1 - (b_{j_1} + \mathsf{Z} \cdot \mathsf{R}_{X,j_1}(\mathsf{Z}))(b_{j_2} + \mathsf{Z} \cdot \mathsf{R}_{X,j_2}(\mathsf{Z})) \qquad (4.4) \\
&= 1 - b_{j_1}b_{j_2} + \mathsf{Z} \cdot \underbrace{(-b_{j_1}\mathsf{R}_{X,j_2}(\mathsf{Z}) - b_{j_2}\mathsf{R}_{X,j_1}(\mathsf{Z}) + \mathsf{Z}\mathsf{R}_{X,j_1}(\mathsf{Z})\mathsf{R}_{X,j_2}(\mathsf{Z}))}_{:=\mathsf{R}_{X,j}(\mathsf{Z})} \\
&= b_j + \mathsf{Z} \cdot \mathsf{R}_{X,j}(\mathsf{Z})
\end{aligned}$$

Note that Equation (4.4) holds because we have by induction that $\mathsf{P}_{X,j_1}(\mathsf{Z}) = b_{j_1} + \mathsf{Z} \cdot \mathsf{R}_{X,j_1}(\mathsf{Z})$ and $\mathsf{P}_{X,j_2}(\mathsf{Z}) = b_{j_2} + \mathsf{Z} \cdot \mathsf{R}_{X,j_2}(\mathsf{Z})$ holds.

Moreover, notice that for $d_j$, the depth of the gate with index $j$, it holds that $d_j = 1 + \max\{d_{j_1}, d_{j_2}\}$, where $d_{j_1}$ and $d_{j_2}$ are the depths of the gates with index $j_1$ and $j_2$ respectively. We then have that

$$\begin{aligned}
\mathsf{deg}(\mathsf{P}_{X,j}(\mathsf{Z})) &= \mathsf{deg}(1 - \mathsf{P}_{X,j_1}(\mathsf{Z})\mathsf{P}_{X,j_2}(\mathsf{Z})) = \mathsf{deg}(\mathsf{P}_{X,j_1}(\mathsf{Z})) + \mathsf{deg}(\mathsf{P}_{X,j_2}(\mathsf{Z})) \\
&= 2^{d_{j_1}} + 2^{d_{j_2}} \leq 2 \cdot 2^{\max\{d_{j_1}, d_{j_2}\}} = 2^{d_j}
\end{aligned}$$

□

Finally, we provide the proof for Lemma 12.

*Proof of Lemma 12.* On input $(\mathcal{BG}, g, h, g^\alpha, \ldots, g^{\alpha^q}, T)$, where $T$ is either $e(g,h)^{1/\alpha}$ or a random element in $\mathbb{G}_T$, the algorithm $\mathcal{B}$ samples $\tilde{w}_0 \xleftarrow{\$} \mathbb{Z}_p^*$ and $\tilde{w}_j \xleftarrow{\$} \mathbb{Z}_p^*$ for all $i \in \mathcal{S}$. It further samples $\mathsf{K}^{\mathsf{part}} \xleftarrow{\$} \mathsf{PrtSmp}(1^\lambda, Q)$ and $\mathsf{K}^{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^m$. For all $j \in \mathcal{S}$ it then sets

$$W_j := \begin{cases} (g^\alpha)^{\tilde{w}_j}\, g^{\mathsf{K}^{\mathsf{PRF}}_{j-|\mathcal{P}|}} & \text{if } j \in \mathcal{S}^{\mathsf{PRF}} \text{ and} \\ (g^\alpha)^{\tilde{w}_j}\, g^{\mathsf{K}^{\mathsf{part}}_{j-|\mathcal{P}|-|\mathcal{S}^{\mathsf{PRF}}|}} & \text{if } j \in \mathcal{S}^{\mathsf{part}}. \end{cases}$$

Further, $\mathcal{B}$ sets $W_0 := (g^\alpha)^{\tilde{w}_0}$. It then gives $\mathsf{vk} := (\mathcal{BG}, g, h, W_0, (W_j)_{j \in \mathcal{S}})$ to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a query $X^{(i)}$ to $\mathsf{Eval}_{\mathsf{opt}}$, then $\mathcal{B}$ computes the coefficients of the polynomials $\mathsf{P}_{X^{(i)},j}(\mathsf{Z})$ for all $j \in \mathcal{C}$. Note that by Lemma 11, we have that the coefficient for degree zero of $\mathsf{P}_{X^{(i)},\mathsf{out}}(\mathsf{Z})$ is identical to $\mathsf{G}(X^{(i)}, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}})$. Hence, if the coefficient of degree zero is 1 for any query $X^{(i)}$, then $\mathcal{B}$ aborts and outputs a random bit just as the challenger in Game 6 and Game 7.

Otherwise, $\mathcal{B}$ computes $Y$ and $\pi$ as

$$Y \coloneqq e\left(g^{\mathsf{R}_X(\alpha)/\tilde{w}_0}, h\right), \qquad \pi \coloneqq \left(\pi_0 = g^{\mathsf{R}_X(\alpha)/\tilde{w}_0}, \left(\pi_j \coloneqq g^{\mathsf{P}_{X^{(i)},j}(\alpha)}\right)_{j\in\mathcal{C}}\right).$$

Note that $\mathcal{B}$ can compute these values because all $\mathsf{P}_{X^{(i)},j}(\mathsf{Z})$ and $\mathsf{R}_{X^{(i)}}(\mathsf{Z})$ have degree at most $2^d \leq q$ and therefore all group elements $g^{x^i}$ for $i \leq 2^d$ are part of the $q$-$\mathsf{DBDHI}$ instance.

When $\mathcal{A}$ submits its challenge $X^*$, $\mathcal{B}$ computes the coefficients of $\mathsf{R}^*_X(\mathsf{Z})$ and $\mathsf{P}_{X^*,j}(\mathsf{Z})$ for all $j \in \mathcal{C}$ as above. As the challenger in Game 6 and Game 7, $\mathcal{B}$ aborts and outputs a random bit if the coefficient of degree zero of $\mathsf{P}_{X^*,\mathsf{out}}(\mathsf{Z})$ is not 1, *i.e.*, if $\mathsf{G}(X^*, \mathsf{K}^{\mathsf{PRF}}, \mathsf{K}^{\mathsf{part}}) = 0$. Otherwise, $\mathcal{B}$ draws a random bit $b \xleftarrow{\$} \{0,1\}$ and returns

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

to $\mathcal{A}$ if $b = 0$ and $\mathcal{B}$ returns $Y^* \xleftarrow{\$} \mathbb{G}_T$ to $\mathcal{A}$ if $b = 1$.

In order to conclude the proof, observe that $\pi$ and $Y$ for all evaluation queries and $Y^*$ for the challenge are distributed exactly as in Game 6 if $T = e(g,h)^{1/\alpha}$. Analogously, if $T \xleftarrow{\$} \mathbb{G}_T$, then $\pi$ and $Y$ for all evaluation queries and $Y^*$ for the challenge are distributed exactly as in Game 7 and therefore

$$\varepsilon_{\mathcal{B}} \coloneqq \mathsf{Adv}^{q\text{-}\mathsf{DBDHI}}_{\mathcal{B}}(\lambda) = |\Pr[\mathsf{G}_6] - \Pr[\mathsf{G}_7].|$$

Furthermore, observe that $t_{\mathcal{A}} \approx t_{\mathcal{B}}$ because $t_{\mathcal{A}}$ already includes the runtime of the security experiment and $\mathcal{B}$ does nothing more than executing the security experiment for $\mathcal{B}$ with the sole difference that it has to compute the coefficients of the polynomials $\mathsf{P}_{X,j}(\mathsf{Z})$ and $\mathsf{R}_X(\mathsf{Z})$. However, these few additional operations do not make a significant difference in the overall runtime of $\mathcal{B}$. □

## 4.5 Conclusion and Open Problems

We showed that every reduction from a non-interactive complexity assumption to the pseudorandomness of a VRF that can sequentially rewind the adversary a constant number of times necessarily loses a factor of $\approx Q$. This settles the question of the optimal tightness an adaptively-secure VRF can achieve under a non-interactive complexity assumption. Furthermore, we constructed the first VRF with a reduction that has this optimal tightness. The takeaway message is that the optimal loss for adaptively-secure VRFs is $Q$ and that it is possible to construct VRFs that attain this bound.

Our main technical contributions are:

1. The extension of the lower bound for the loss of reductions by Bader *et al.* [BJLS16] to VRFs and VUFs in Section 4.3.

2. Further, we presented a new partitioning strategy that achieves this optimal tightness even in the context of decisional security notions and complexity assumptions.

3. Finally, we show that this partitioning strategy can be applied in Yamada's VRF and thus yields a VRF in the standard model with optimal tightness. This also shows that the lower bound on the loss of reductions from a non-interactive complexity assumption to the security of a VRF that we present is optimal.

However, there are still some open questions. The technique of Bader *et al.*, and therefore also our results, only applies to non-interactive complexity assumptions and reductions that sequentially rewind adversaries. While this result covers already a large class of assumptions and reductions, it does not cover interactive assumptions and reductions that can run several instances of the adversary in parallel. Morgan and Pass show a lower bound of $\sqrt{Q}$ for the loss of reductions to the unforgeability of unique signatures from interactive assumptions [MP18]. While it seems plausible that their technique could be extended to also cover VRFs and VUFs, this is still an open question.

**Research Question 3.** *Is there a lower bound on the reduction loss of proofs of pseudorandomness or unpredictability when the reduction is from an interactive complexity assumptions or the reduction can execute several instances of the adversary at the same time? In particular, can the technique by Morgan and Pass [MP18] be extended to also cover VUFs or VRFs?*

Moreover, $\mathcal{VRF}_{\mathsf{opt}}$ is less efficient than the VRFs that we discussed in Chapter 3. This raises the question whether a comparable efficiency can be achieved together with tightness.

**Research Question 4.** *Are there verifiable random functions that can be proven secure with optimal tightness loss and are as efficient as currently known VRFs without optimal tightness loss?*

Similar to the VRFs that we introduced in Chapter 3, $\mathcal{VRF}_{\mathsf{opt}}$ is based on the $q$-DBDHI assumption with a polynomial $q$, which is not a standard assumption and gets stronger with $q$ [Che10]. It would therefore be preferable to construct an efficient VRF with optimal tightness from a standard assumption, like the VRFs in [HJ16, Koh19, Ros18].

**Research Question 5.** *Are there verifiable random functions that can be proven secure under a standard assumption and achieve optimal tightness?*

# 5 Efficient Identity-Based Key-Encapsulation Schemes from Lattices

In this chapter, we construct new *programmable hash function*s (PHFs) for lattices. The notion of PHFs was introduced by Hofheinz and Kiltz [HK08, HK12], and it provides a modular building block to construct efficient cryptosystems with adaptive-security in the standard model. Zhang *et al.* first introduced PHFs specifically for lattices in [ZCZ16]. We introduce balanced PHFs for lattices that, in contrast to standard PHFs, enable cryptographic schemes with decisional security, such as IB-KEMs, without requiring additional techniques such as the artificial abort technique by Waters [Wat05]. We show a construction of a lattice-based IB-KEM that is secure in the standard model and uses balanced PHFs in a black-box manner. Furthermore, since balanced PHFs for lattices also fulfill all requirements for standard PHFs for lattices, the digital signature schemes by Zhang *et al.* [ZCZ16] can be instantiated with all balanced PHFs that we present.

Overall, we present three balanced PHFs. Our first balanced PHF uses the blockwise partitioning technique we introduced in Section 3.4. Thus, it requires weak near-collision resistant hash functions for its security. It has a description that essentially consists of $\mathcal{O}(\log \lambda)$ many LWE matrices and thus gives rise to the currently most efficient IB-KEM from lattices with a polynomial reduction loss and polynomial LWE parameters.

Even though we view weak near-collision resistance of hash functions as a natural assumption for standardized hash functions, it would still be preferable to base security on well-studied complexity assumptions. We, therefore, define *exponentially-collision resistant* (ECR) hash functions as a weaker assumption for hash functions than the assumptions we used in Chapter 3. We then concretely construct ECR hash functions from the exponential SIS assumption, which extends the standard SIS assumption with a concrete bound on the running time of algorithms. Thus, in contrast to TCR (see Definition 20) and wNCR (see Definition 21), we prove the existence of ECR hash functions by assuming the hardness of a well-studied computational problem.

We then demonstrate the potential of ECR hash functions by presenting two balanced PHFs from ECR hash functions that achieve even shorter function descriptions than the PHF based on blockwise partitioning. The first of these PHFs has a function description consisting essentially of a *constant* number of matri-

ces and gives rise to the first lattice-based IB-KEM with adaptive security in the standard model, but with a slightly super-polynomial LWE parameter. Our last balanced PHF achieves a polynomial LWE parameter with a function description consisting of $\mathcal{O}(\sqrt{\log \lambda})$ many matrices. However, this efficiency comes at the price of a larger reduction loss: Both balanced PHFs from ECR hash functions require a guessing argument, which incurs a "sub-quasi-polynomial" loss of $\lambda^{-\mathcal{O}(\log\log(\lambda))}$ if instantiated with the ECR hash function we present.

This yields very efficient IB-KEMs and signatures, including the first IND-ID-CPA secure lattice-based IB-KEMs with essentially a *constant* number of matrices in their master public keys and adaptive security. Moreover, it yields the most efficient IB-KEM with polynomially-bounded LWE parameter. Finally, it also yields a signature scheme, which directly achieves full EUF-CMA security with constant-size verification key, and the most efficient signature scheme with polynomially-bounded SIS parameter secure under SIS.

Author's contributions. The results presented in this chapter are based on joined work with Tibor Jager and Rafael Kurek published as [JKN21] and on unpublished joined work with Tibor Jager that is to be submitted to PKC 2022. The formulation of balanced PHFs as an extension to plain PHFs and the definition and construction of the ECR hash functions are the result of intensive discussions with Tibor Jager, and both contributed equally. Apart from this, the author made the following technical contributions:

- The construction of balanced PHFs from blockwise partitioning, which is published in [JKN21];
- The formulation and proof of the bound on the size of the description of ECR hash functions of exponentially increasing output length presented in Lemma 21;
- The construction and the accompanying proofs of the PHFs from ECR functions that we present in Section 5.5 and Section 5.6;
- The construction and the accompanying proofs of the IB-KEM presented in Section 5.7.2, which is a variation of the identity-based encryption scheme by [Yam17a, Section 5].

## 5.1 Motivation and Overview

The generality of PHFs as an abstract building block makes it possible to "outsource" a partitioning argument, as we describe it in Section 2.6, from a security proof to the PHF. This makes security proofs more modular, it reduces their complexity significantly, and avoids a tedious repetition of standard arguments. Furthermore, it makes it possible to instantiate schemes with different PHFs without the need for a new security analysis.

Lattice-based PHFs. Various different types of PHFs were constructed in several follow-up works [HK08, HK12, HJK11, FHPS13, CFN15, ZCZ16]. Most relevant to this chapter is the adoption to the setting of lattice-based cryptography by Zhang *et al.* [ZCZ16], who constructed the first lattice-based PHFs and showed how to use them to construct short signatures and *identity-based encryption* (IBE) schemes with small key sizes based on lattice assumptions.

A lattice-based PHF is based on a *matrix hash function* $\mathcal{F} = (\mathsf{HGen}, \mathsf{HEval})$, where $\mathsf{HGen}(1^\lambda)$ takes the security parameter $\lambda$ and produces a function description $F$, and $\mathsf{HEval}(F, X)$ evaluates the function $X \mapsto F(X) \in \mathbb{Z}_q^{n \times m}$. Essentially, $\mathcal{F}$ is a PHF if there exist additional efficient algorithms $\mathsf{TrapGen}$ (*trapdoor key generation*) and $\mathsf{TrapEval}$ (*trapdoor evaluation*) with the following properties.

1. $\mathsf{TrapGen}(\mathbf{A})$ takes a matrix $\mathbf{A}$ (e.g., from an LWE instance) and produces a function description $F'$, which is indistinguishable from one generated by $\mathsf{HGen}$, along with a trapdoor $\mathsf{td}$.

   Intuitively, this makes it possible to replace $\mathsf{HGen}$ with $\mathsf{TrapGen}(\mathbf{A})$ in a security proof, such that an instance $\mathbf{A}$ of a hard lattice problem is embedded into the function description and that the reduction knows some additional trapdoor information $\mathsf{td}$.

2. $\mathsf{TrapEval}(\mathsf{td}, X)$ produces two matrices $(\mathbf{R}_X, \mathbf{H}_X)$ such that one can write $\mathsf{HEval}(F', X)$ as

   $$\mathsf{HEval}(F', X) = \mathbf{A}\mathbf{R}_X + \mathbf{H}_X\mathbf{G} \tag{5.1}$$

   where $\mathbf{G}$ is the so-called *gadget matrix* that allows the reduction to solve LWE or SIS if the norm of $\mathbf{R}_X$ is sufficiently small. We formally introduce $\mathbf{G}$ and its properties in Lemma 13.

   Intuitively, this ensures that the reduction is able to use the trapdoor to obtain a representation of the form from Equation (5.1) of the output of $\mathsf{HEval}(F', X)$, which can be used with standard lattice proof techniques to simulate a security experiment whenever $\mathbf{H}_X \neq \mathbf{0}$.

3. For any sequence of inputs $\mathcal{X} = (X_1, \ldots, X_Q, X^*)$ holds

   $$\mathbf{H}_{X^*} = \mathbf{0} \qquad \text{and} \qquad \mathbf{H}_{X_i} \neq \mathbf{0} \,\forall\, i \in \{1, \ldots, Q\}$$

   with some non-negligible probability.

   Intuitively, this *"well-distributedness"* requirement enables a "partitioning argument" in a security proof. It ensures that we have $\mathbf{H}_X = \mathbf{0}$ *only* for $X = X^*$. For example, for identity-based encryption this will make it possible to embed an instance of a hard problem into a ciphertext associated to "identity" $X^*$, whereas for signatures this will enable the extraction of a solution to a hard problem from a forgery for "message" $X^*$

Limitations of known lattice-based PHFs. The known constructions of lattice-based PHFs by Zhang *et al.* [ZCZ16] enable the generic and modular application of PHFs to construct cryptosystems like IBE and signature schemes from standard lattice assumptions. However, they still exhibit the following limitations, which we seek to address.

**Efficiency and dependence on $Q$.** For typical applications of PHFs, including the IBE and signature schemes in [ZCZ16] and the IB-KEM we present, the function description $F$ is contained in the public key (concretely, in the master public key of an IBE scheme or the public key of a signature scheme). Therefore we measure the efficiency of a PHF by the size $|F|$ of the function description.

For the second lattice-based PHF construction from [ZCZ16], $F$ consists of $\log(Q)$ matrices in $\mathbb{Z}_q^{n\times m}$. While it asymptotically holds that $\log(Q) = O(\log\lambda)$ for a polynomial $Q = Q(\lambda)$, any concrete instantiation of $F$ still depends on $Q$. For instance, for the applications of PHFs in [ZCZ16] and our work, $Q$ corresponds to the number of identity key queries made by an adversary. Therefore the resulting schemes only achieve security against adversaries that ask up to $Q$ queries ("$Q$-bounded security"), but no security against arbitrary polynomial-time adversaries which might ask $Q+1$ or more queries.

Furthermore, this construction seems rather impractical, unless $Q$ is very small, since it is based on $Q$-cover free sets.

The only other currently known PHF for lattices is the first construction from [ZCZ16]. However, this construction requires $O(\lambda)$ many matrices in $|F|$.

**Dependence on $\mathcal{X}$, the "artificial abort", and modularity.** Note that the *"well-distributedness"* of a PHF guarantees only that a partitioning argument succeeds with non-negligible probability. However, it does not guarantee that the partitioning works (or fails) with approximately the same probability for all possible sequences of queries $\mathcal{X} = (X_1, \ldots, X_Q, X^*)$, as required when reducing from a decisional assumption as we discussed in Section 2.6.

While the standard approach of performing an *artificial abort*, which was also used by Zhang *et al.* [ZCZ16], resolves this issue, it comes at the price of a significant computational overhead. Furthermore, it adds significant complexity to reductions and redundancy to proofs.

To reduce the computational overhead, Bellare and Ristenpart [BR09a] showed how to avoid artificial aborts. However, their analysis is not generically based on PHFs, but more "low-level" and therefore not modular. This makes it difficult to generically and modularly apply such PHFs for the construction of cryptosystems without requiring artificial abort.

Contributions. We make progress regarding the issues mentioned above by introducing balanced PHFs and demonstrating how to construct an IB-KEM from a balanced PHF in a black-box manner without relying on the artificial abort technique. We then present three balanced PHFs, one whose security relies on wNCR hash functions and two balanced PHFs that essentially rely on the exponential hardness of the well-studied *short integer solution* (SIS) problem. We discuss the contributions in more detail below.

- We define *balanced programmable hash function*s (balanced PHFs) as an extension of plain PHFs. Balanced PHFs are "balanced" in the sense that they do not require the artificial abort technique in order to prove the security of IB-KEMs or other primitives with indistinguishability-based security definitions. Furthermore, for all three constructions of balanced PHFs we present, the size $|F|$ of the function description is independent of $Q$.

- Our first PHF construction ($\mathcal{F}_{\mathsf{BLK}}$) is based on the semi-generic blockwise partitioning technique (see Lemma 6) and has a function description $F$ that consists of $\mathcal{O}(\log \lambda)$ many matrices. This construction requires wNCR hash functions (see Definition 21) in order to be secure. It currently allows for the most efficient constructions of IB-KEMs with polynomial LWE parameters and a polynomial reduction loss.

- Even though we deem wNCR a plausible assumption, it is still desirable to replace it with a well-studied computational problem. To that end, we introduce *exponentially-collision resistant* (ECR) hash functions, which essentially require that any algorithm for finding collision takes exponential time in the *output length*. On a high level, this approach follows the same paradigm as wNCR and TCR in the sense that it makes it possible to instantiate a hash function $H : \{0,1\}^* \to R$ such that $R$ is "sufficiently small" to make it possible to guess $H(X^*)$ with significant success probability, while at the same time $R$ is "sufficiently large" that it is possible to argue that an efficient adversary must (sometimes) be able to break the cryptosystem without finding a collision. We provide a construction of ECR hash functions based on lattice assumptions. We show how to construct ECR hash functions based on lattice assumptions explicitly. To this end, we define the *exponential SIS assumption* (eSIS), which essentially extends the standard SIS assumption [Ajt96] with a concrete bound on the success probability of any adversary running within a certain amount of time. In our construction, the set $R$ is of size $\lambda^{\mathcal{O}(\log\log(\lambda))}$, which is super-polynomial but still sub-quasi-polynomial. Hence, our reduction can only guess $H(X^*)$ correctly with probability $\lambda^{-\mathcal{O}(\log\log(\lambda))}$, thus losing a quasi-polynomially bounded factor.

- We demonstrate the potential of exponentially-collision resistant hash functions by presenting two balanced PHFs from ECR hash functions. Both constructions use $\ell = \Theta(\log \lambda)$ ECR hash functions $H_1, \ldots, H_\ell$ in parallel,

such that there exists a suitable choice $H_i$ for every runtime $t$ and advantage $\varepsilon$ an adversary might have. Our first balanced PHF from ECR hash functions has a function description $F$, which *essentially* consists of a *constant* number of matrices. More precisely, it consists of two matrices in $\mathbb{Z}_q^{n\times m}$ and $\ell = O(\log \lambda)$ keys for hash functions, where $\lambda$ is the security parameter. However, the hash function keys require only space $O(|\mathbb{Z}_q^{n\times m}|)$, where $|\mathbb{Z}_q^{n\times m}|$ denotes the size of one matrix in $\mathbb{Z}_q^{n\times m}$. This construction extends a technique by Alperin-Sheriff [Alp15], which was used to construct efficient lattice-based signatures but only achieved *non-adaptive* security. We show how to leverage this approach to construct PHFs, which then directly yields *adaptive* security. The main downside of this construction is that it only achieves a (slightly) super-polynomial upper bound on the norm of the trapdoor matrices and thus also requires modulus $q$ of (slightly) super-polynomial size.

- Our second balanced PHF construction ($\mathcal{F}_{\mathsf{SN}}$) addresses this downside. Here, a function description again consists of $\ell = O(\log \lambda)$ keys for hash functions (which again can be compactly represented using space $O(|\mathbb{Z}_q^{n\times m}|)$), along with only $O(\sqrt{\ell}) = O(\sqrt{\log \lambda})$ $\mathbb{Z}_q^{n\times m}$-matrices. Essentially, the idea of this construction is to represent $\ell$ matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ that correspond to the "actual" function description by only $2\sqrt{\ell}$ matrices. We achieve this by following a baby-step giant-step approach that we describe in more detail in Section 5.6.

- Balanced PHFs can not immediately be used in the IBE scheme by Zhang *et al.* [ZCZ16] because it requires an additional "min-entropy" property from the PHF. Therefore, we describe a new IB-KEM, which does not have this min-entropy requirement and can be proven secure under LWE by using a balanced PHF in a black-box manner. We compare the properties of our IB-KEM with lattice-based IBEs schemes in Table 5.2 in Section 5.7, which is based on the respective table by Yamada in [Yam17a, Table 1].

Application to lattice-based signatures. All three of our PHFs directly give rise to more efficient digital signature schemes by plugging the PHF into the digital signature construction from [ZCZ16].

$\mathcal{F}_{\mathsf{AS}}$ gives rise to a signature scheme where the public parameters essentially consist of a *constant* number of matrices. This approach is similar to the construction by Alperin-Sheriff [Alp15], which, achieves only *non-adaptive* security and thus requires an additional chameleon hash function or a one-time signature to achieve adaptive security which increases the size of signatures. Ours directly achieves adaptive security.

When instantiated with $\mathcal{F}_{\mathsf{SN}}$, we obtain the most efficient signature scheme secure under the *short integer solution* (SIS) problem with polynomially bounded SIS parameter $\beta$. We compare ours and previous signature schemes in Table 5.1, which is based on the respective table from [KONT20]. Note that Table 5.1 lists schemes from SIS and RingSIS to give a better overview over lattice-based digital signature

schemes in the standard model, even though our claim is only subject to digital signature schemes secure under the standard SIS problem. Moreover, it is plausible that our techniques could also be applied to constructions of digital signatures secure under RingSIS. However, we do not make any claims regarding this.

Application to lattice-based IB-KEMs. Unfortunately, our PHFs cannot immediately be used in the IBE scheme by Zhang *et al.* [ZCZ16] because their IBE scheme requires an additional "min-entropy" property from the PHF. Therefore we describe a new IB-KEM, which does not require the PHFs to fulfill the "min-entropy" property, but otherwise has similar performance when instantiated with the same PHF.

Given the modularity of the PHF abstraction, we can base this construction in a relatively straightforward way on the construction by Yamada [Yam17a]. For simplicity, we build an *identity-based key encapsulation mechanism* (IB-KEM), which, however, directly gives rise to an IBE scheme (since we consider IND-ID-CPA security, see Definition 8). The *balancedness* allows us to avoid an artificial abort. We provide a detailed comparison of our schemes with previous ones in Table 5.2, which is based on the respective tables in [Yam17a].

For constructions with a polynomial LWE parameter, Table 5.2 shows that if our IB-KEM is instantiated with $\mathcal{F}_{\mathsf{BLK}}$, our balanced PHF based on blockwise partitioning, which we present in Section 5.3, it achieves master public keys consisting of only $\mathcal{O}(\log \lambda)$ many matrices but requires us to assume the existence of wNCR hash functions. This already improves over previous lattice-based IB-KEMs and IBEs. Without non-standard assumptions, the schemes of Yamada [Yam17a] from CRYPTO'17 are the most efficient ones since they achieve master public keys that consist of only $\log^2(\lambda)$ and $\log^3(\lambda)$ many matrices, respectively. Note that the scheme by Apon *et al.* [AFL17] achieves a constant number of matrices in the master public key but requires that the matrix dimension $m$ is at least $\mathcal{O}(\lambda^{1+\delta})$ for some $\delta > 0$. This makes the master public keys asymptotically larger than those of the previous schemes, which all only require $m = \mathcal{O}(\lambda \log \lambda)$. In contrast to these previous constructions, when instantiated with $\mathcal{F}_{\mathsf{SN}}$, our IB-KEM is the most efficient IB-KEM with polynomially bounded LWE parameter $1/\alpha$, since it essentially requires only $\mathcal{O}(\sqrt{\log(\lambda)})$ many matrices in the master public key. When instantiated with $\mathcal{F}_{\mathsf{AS}}$, we obtain the first IB-KEM with full adaptive IND-ID-CPA security, which can be instantiated with essentially a *constant*-size (in the number of matrices) master public key.

Note that we say "*essentially*" when counting the matrices in the master public key of our construction since the description of each of the PHFs contains the description logarithmically many ECR hash functions. However, we show that these logarithmically many ECR hash functions can alltogether be described by what is equivalent to $\mathcal{O}(1)$ many matrices from $\mathbb{Z}_q^{n \times m}$.

| **Scheme** | $\lvert\mathsf{vk}\rvert$ | $\lvert\mathsf{sk}\rvert$ | $\lvert\mathsf{sig}\rvert$ | Reduction loss | $\beta$ |
|---|---|---|---|---|---|
| [Boy10] | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(nQ)$ | $\Omega(n^2)$ |
| [CHKP12] | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(nQ)$ | $\Omega(n^{7/2})$ |
| † [BHJ+15] | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(\log(n))$ | $\mathcal{O}\left(Q^{2\nu}/\varepsilon^{\nu}\right)$ | $\Omega(n^{5/2})$ |
| ✠ † [DM14] | $\mathcal{O}(\log n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}\left(Q^{2\nu}/\varepsilon^{\nu}\right)$ | $\Omega(n^{7/2})$ |
| † [BKKP15] | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | $\Omega(n^{3/2})$ |
| † [Alp15] | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}\left(Q^{2\nu}/\varepsilon^{\nu}\right)$ | $\mathsf{superpoly}(\lambda)$ |
| [BL16] | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(n)$ | $\Omega(n^{7/2}\zeta^{4\mu})$ |
| $\mathcal{SIG}_1$ in [ZCZ16] | $\mathcal{O}(\log n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(nQ^2)$ | $\Omega(Q^2n^{11/2})$ |
| $\mathcal{SIG}_2$ in [ZCZ16] | $\omega(\log n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(nQ)$ | $\Omega(n^{11/2})$ |
| ✠ Scheme 1 in [KONT20] | $\mathcal{O}(\log n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}\left(Q^{\nu}/\varepsilon^{\nu}\right)$ | $\Omega(n^{7/2})$ |
| ✠ Scheme 2 in [KONT20] | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(Q/n)$ | $\Omega(n^{7/2})$ |
| Signatures from $\mathcal{F}_{\mathsf{blk}}$ | $\mathcal{O}(\log \lambda)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(t^2/\varepsilon^2)$ | $\Omega(n^{7/2})$ |
| Signatures from $\mathcal{F}_{\mathsf{AS}}$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\lambda^{\mathcal{O}(\log\log(\lambda))}$ | $\mathsf{superpoly}(\lambda)$ |
| Signatures from $\mathcal{F}_{\mathsf{SN}}$ | $\mathcal{O}\left(\sqrt{\log(\lambda)}\right)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\lambda^{\mathcal{O}(\log\log(\lambda))}$ | $\Omega(n^{7/2})$ |

*Table 5.1:* EUF-CMA secure signatures based on (Ring-)(I)SIS in the standard model

Above, we compare EUF-CMA secure digital signature schemes based on the SIS assumption and well established variants of it like ISIS and Ring-SIS. $\lvert\mathsf{vk}\rvert$, $\lvert\mathsf{sk}\rvert$ and $\lvert\mathsf{sig}\rvert$ respectively denote the size of the verification key, the secret key and the signatures, where the size of $\mathsf{vk}$ is measured in the number of matrices it consists of and we count the number of vectors for the latter two. The lattice dimension is $n = \Theta(\lambda)$. For the parameters, we have $t, Q \in \mathbb{N}$ and $\varepsilon \in [0, 1]$ are, respectively, the runtime of the adversary, the number of signature queries made by the adversary and its advantage. The parameter $\nu > 0$ is a constant that can be chosen arbitrarily and is common to signatures employing the confined guessing approach [BHJ+15]. Furthermore, the following notes apply.

- The parameters for signatures from $\mathcal{F}_{\mathsf{BLK}}$, $\mathcal{F}_{\mathsf{AS}}$ and $\mathcal{F}_{\mathsf{SN}}$ refer to the instantiation of $\mathcal{SIG}_1$ from [ZCZ16] with our respective PHF and then applying Theorem 6 from [ZCZ16].
- The signature scheme by [BL16] requires a PRF that can be evaluate by a NAND-circuit of logarithmic depth and polynomial size. $\zeta$ is the number of input bits of the circuit and $\mu$ is chosen such that $\mu \log(\zeta)$ is an upper bound on the depth of the circuit.
- Schemes marked with † need to employ a chameleon hash function in order to achieve adaptive security. This requires a constant number of further matrices in the verification key [CHKP12, Section 4.1].
- Schemes marked by ✠ are based on the hardness of the SIS problem over rings instead of standard lattices and hence the size of the verification key refers to vectors instead matrices.

| Scheme | $\lvert\mathsf{mpk}\rvert$ # of $\mathbb{Z}_q$ elements | $\lvert\mathsf{usk}\rvert, \lvert\mathsf{ct}\rvert$ # of $\mathbb{Z}_q^m$ vec. | LWE param $1/\alpha$ | Reduction Cost | Remarks |
|---|---|---|---|---|---|
| [CHKP10] | $\mathcal{O}(\lambda \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(\lambda)$ | $\tilde{\mathcal{O}}(n^{1.5})$ | $\mathcal{O}(\varepsilon^{\nu+1}/Q^{\nu})^{\ddagger}$ | |
| [ABB10b]+[Boy10] | $\mathcal{O}(\lambda \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(n^{5.5})$ | $\mathcal{O}(\varepsilon^2/qQ)$ | |
| [Yam16] | $\mathcal{O}(\lambda^{1/\mu} \cdot \lambda^2 \cdot \log(\lambda))^{\dagger}$ | $\mathcal{O}(1)$ | $n^{\omega(1)}$ | $\mathcal{O}(\varepsilon^{\mu+1}/kQ^{\mu})^{\dagger}$ | |
| [ZCZ16] | $\mathcal{O}(\log(Q) \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(Q^2 n^{6.5})$ | $\mathcal{O}(\varepsilon/kQ^2)$ | $Q$-bounded |
| [AFL17]$^{*}$ | $\mathcal{O}(\lambda^2 \cdot \lambda^{\delta})$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(n^6)$ | $\mathcal{O}(\varepsilon^2/qQ)$ | |
| [BL16] | $\mathcal{O}(\lambda \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\mathsf{superpoly}(n)$ | $\mathcal{O}(\lambda)$ | |
| [Yam17a] + $\mathsf{F}_{\mathsf{MAH}}$ § | $\mathcal{O}(\log^3(\lambda) \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(n^{11})$ | $\mathcal{O}(\varepsilon^{\nu+1}/Q^{\nu})^{\ddagger}$ | |
| [Yam17a] + $\mathsf{F}_{\mathsf{AFF}}$ § | $\mathcal{O}(\log^2(\lambda) \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\mathsf{poly}(\lambda)$ | $\mathcal{O}(\varepsilon^2/k^2Q)$ | Expensive offline phase |
| Constr. 8 + Sec. 5.3 | $\mathcal{O}(\log(\lambda) \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(n^6)$ | $\mathcal{O}(\varepsilon^2/t^2)$ | Needs wNCR hash functions |
| Constr. 8 + Sec. 5.5 | $\mathcal{O}(\lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\mathsf{superpoly}(n)$ | $\lambda^{\mathcal{O}(\log\log(\lambda))}$ | Needs eSIS |
| Constr. 8 + Sec. 5.6 | $\mathcal{O}(\sqrt{\log(\lambda)} \cdot \lambda^2 \cdot \log(\lambda))$ | $\mathcal{O}(1)$ | $\tilde{\mathcal{O}}(n^{13})$ | $\lambda^{\mathcal{O}(\log\log(\lambda))}$ | Needs eSIS |

*Table 5.2:* Adaptively-secure IBEs based on LWE in the standard model
The table above compares the relevant parameters of previous LWE based IBEs in the standard model with instantiations of our IB-KEM. We measure the size of $\mathsf{ct}$ and $\mathsf{usk}$ in the number of $\mathbb{Z}_q^m$ vectors and the size of $\mathsf{mpk}$ in the number of $\mathbb{Z}_q$ elements. By $Q \in \mathbb{N}, \varepsilon \in [0,1]$ and $t \in \mathbb{N}$ we denote the number of queries, the advantage against the security of the respective IBE, and the runtime of an adversary, respectively. The reduction cost is computed as the advantage that the reduction has in solving the LWE problem relative to the advantage of an adversary against the IBE. We compute all reductions costs by using the technique from [BR09a].

† The constant $\mu \in \mathbb{N}$ can be chosen arbitrarily. However, the reduction cost degrades exponentially in $\mu$ and hence it should be chosen rather small.

‡ $\nu > 1$ describes the relative distance $c = 1 - 2^{-1/\nu}$ of a binary error correcting code that is used in these constructions. By applying standard constructions from coding theory, $\nu$ can be chosen arbitrarily close to 1 [Gol08].

$*$ The construction requires $m = n^{1+\delta}$ for some $\delta > 0$. Note that this is asymptotically larger than $m = \mathcal{O}(n \log n)$, which is used in all other constructions listed above. We do not include the prior draft [AFL16] of [AFL17] in the table above, because the latter one completely subsumes former one.

§ Yamada [Yam17a] provides two instantiations of his IBE. The first one is based on a modified admissible hash function ($\mathsf{F}_{\mathsf{MAH}}$) and the second one is based on affine functions ($\mathsf{F}_{\mathsf{AFF}}$). Note that the construction based on $\mathsf{F}_{\mathsf{AFF}}$ requires the key generation and encryption algorithms to compute the description of a branching program that computes the division and thus makes the construction less efficient.

Further related work. Similar to our assumption of the exponential hardness of the SIS problem for appropriate parameters, Zhandry [Zha16, Zha19a, Zha19b] used exponential hardness of the DDH problem in elliptic curve groups to construct *extremely lossy functions* (ELFs), which we already discussed in Chapter 3. Our eSIS assumption can be seen as a "lattice analog" to Zhandry's exponential hardness assumption of the DDH problem.

Balancedness of partitioning proofs was also considered in [Jag15] in the context of admissible hash functions (AHFs) [BB04b, CHKP12, FHPS13], and we discussed it in detail in Section 3.2. Balanced admissible hash functions provide a different way to perform partitioning proofs. However, as we have shown in Section 3.2.3, known constructions of bAHFs come with a significant overhead due to the redundancy that is inherently introduced by the error-correcting codes that all known constructions of AHFs use.

Yamada [Yam17a] describes another way to build partitioning reductions, which can also be used to construct lattice-based IBE schemes and signatures without random oracles. However, the "compatible algorithms" approach used in [Yam17a] is very low-level and seems not to allow for as modular security proofs as with PHFs. Note that the balanced PHF based on blockwise partitioning is published in [JKN21], and there it is presented in the form of compatible algorithms. We state it in the form of a balanced PHF for consistency.

### 5.1.1 Notation and Preliminaries for Lattices

We provide some further notation necessary in the context of lattices. That is, for $\mathbf{A} \in \mathbb{R}^{m \times m'}$ and arbitrary $m, m' \in \mathbb{N}$ we denote with $\|\mathbf{A}\|_\infty$ the largest absolute value of any entry of $\mathbf{A}$ and with $\|\mathbf{A}\|_2$ we denote the matrix norm induced by the Euclidean vector-norm, also referred to as spectral norm. Furthermore, we denote the transpose of $\mathbf{A} \in \mathbb{R}^{m \times m'}$ by $\mathbf{A}^\mathsf{T} \in \mathbb{R}^{m' \times m}$. Finally, for $\mathbf{B} \in \mathbb{R}^{ntimesm'}$, we denote with $[\mathbf{A} \mid \mathbf{B}]$ the matrix in $\mathbb{R}^{(n+m) \times m'}$ obtained by appending the columns of $\mathbf{B}$ to $\mathbf{A}$ to the right in the same order as in $\mathbf{B}$.

Furthermore, we introduce two important Lemmas that we will repeatedly use throughout this chapter. We will provide further preliminaries on lattices throughout the chapter where necessary.

We begin by defining the "gadget" matrix introduced by Micciancio and Peikert [MP12].

**Lemma 13** (Theorem 1 from [MP12])**.** *Let $m \geq n \lceil \log(q) \rceil$, then there is a fixed "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$, such that $\mathbf{G}$ has full rank and there is an efficient algorithm $\mathbf{G}^{-1}$ that on input a matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$ outputs $\mathbf{V} \in \{0,1\}^{m \times m}$ such that $\mathbf{GV} = \mathbf{U}$.*

Note that $\mathbf{G}^{-1}$ is not the inverse of $\mathbf{G}$ but an efficient algorithm. However, this notation captures very well what $\mathbf{G}^{-1}$ achieves, and we thus deem it justified. This gadget matrix has some further useful properties.

**Lemma 14** (Section 3.4 in [MP12]). *Let* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{R} \in \mathbb{Z}_q^{m\times m}$, *and let* $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ *be an invertible matrix. Then we can efficiently sample from* $[\mathbf{A} \mid \mathbf{AR} + \mathbf{HG}]_\sigma^{-1}$ *for* $\sigma = m \cdot \|\mathbf{R}\|_\infty \cdot \omega(\sqrt{\log(m)})$.

Furthermore, we introduce the *left-over hash lemma*. Informally, the lemma guarantees that for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} \xleftarrow{\$} \{-1, 1\}^{m\times m}$ it holds that $\mathbf{AR}$ is indistinguishable from an independent uniformly random matrix $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ .

**Lemma 15** (Leftover hash lemma [Reg05, ABB10a]). *Let* $m \geq (n+1)(\log(q) + \omega(\log(n))$, *then for* $\mathbf{R} \xleftarrow{\$} \{-1, 1\}^{m\times m}, \mathbf{A}, \mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ *it holds that the statistical difference between the distributions* $(\mathbf{A}, \mathbf{AR})$ *and* $(\mathbf{A}, \mathbf{A}')$ *is negligible in* $n$.

Finally, we introduce the Bertrand–Chebyshev theorem, which will be useful when considering concrete parameters for lattice based schemes. Informally, it states that for any natural number $n$, there will always be a prime number $p$ between $n$ and $2n$. We refer to [MM13] for a proof of the theorem in english.

**Theorem 9** (Bertrand–Chebyshev theorem [Ber45, Che52]). *Let* $n \in \mathbb{N}$ *with* $n > 3$. *Then there is always a prime* $p \in \mathbb{N}$ *with*

$$n < p < 2n.$$

## 5.2 Balanced Programmable Hash Functions for Lattices

In this section, we formally introduce lattice-based *balanced programmable hash functions* (balanced PHFs), based on lattice-based programmable hash functions by Zhang *et al.* [ZCZ16], which in turn are based on [HK08, HK12]. In contrast to standard lattice-based PHFs, which we informally introduced in Section 5.1 above, lattice-based balanced PHFs can be applied in reductions from decisional hardness assumption immediately. We achieve this by following roughly the same approach as for computational admissible hash functions (see Section 3.3) and blockwise partitioning (see Section 3.4). That is, we define the "well-distributedness"property of balanced PHFs via two events $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}_{\mathsf{phf}}$. The former occurs if an adversary generates two inputs $X \neq X'$ for the balanced PHF such that $\mathbf{H}_X = \mathbf{H}_{X'}$ holds, where $\mathbf{H}_X$ and $\mathbf{H}_{X'}$ are the matrices produced by $\mathsf{TrapEval}$ for the inputs $X$ and $X'$. Analogously, $\mathsf{badChal}_{\mathsf{phf}}$ occurs if $\mathbf{H}_X^* \neq \mathbf{0}$, where $X^*$ is the challenge that the adversary chooses in the security experiment and $\mathbf{H}_X^*$ is the respective matrix generated by $\mathsf{TrapEval}$ for $X^*$. Analogous to cAHFs and blockwise partitioning, we then require that

$$(\varepsilon - \Pr[\mathsf{coll}_{\mathsf{phf}}]) \cdot \Pr[\mathsf{badChal}_{\mathsf{phf}}]$$

is non-negligible. This guarantees that all requirements from standard PHFs is fulfilled but moreover enables reductions to decisional complexity assumptions as

we demonstrate by constructing an IND-ID-CPA-secure IB-KEM in Section 5.7. While the events $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}_{\mathsf{phf}}$ resemble the respective events Chapter 3, they concern the embedding of the partitioning argument used in the construction of the balanced PHF instead of the partitioning argument itself. That is why we indexed them with "$\mathsf{phf}$". We formally define balanced PHFs as follows.

**Definition 31.** Let $\mathcal{X}$ be an arbitrary set and let $\mathcal{T} \coloneqq \mathbb{Z}_q^{n\times m}$ for $n = \Theta(\lambda)$ and $m = \Theta(\lambda \log \lambda)$. Further, let $\mathcal{F} = (\mathsf{HGen}, \mathsf{HEval})$ be a pair of algorithms, where $\mathsf{HGen}(1^\lambda)$ produces a description $F$ of a function and $\mathsf{HEval}(F, X)$ implements a function $F : (\mathcal{X} \to \mathcal{T},\ X \mapsto F(X))$. We say that $\mathcal{F}$ is a *$(\beta, \gamma, \delta)$-balanced programmable hash function* (balanced PHF) if there exists a PPT *trapdoor key generation* algorithm $\mathsf{TrapGen}$ and a deterministic polynomial time *trapdoor evaluation* algorithm $\mathsf{TrapEval}$ such that the following holds.

Given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the public gadget matrix $\mathbf{G}$, $t \in \mathbb{N}$ and $\varepsilon \in (0, 1]$ the following properties hold:

**Syntax:** $\mathsf{TrapGen}$ and $\mathsf{TrapEval}$ have the following syntax.

- Algorithm $(F', \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ takes as input the security parameter $\lambda$, $t \in \mathbb{N}$, $\varepsilon \in [0, 1]$, and a matrix $\mathbf{A}$. It outputs a hash function description $F'$ and a trapdoor $\mathsf{td}$.
- Algorithm $(\mathbf{R}_X, \mathbf{H}_X) = \mathsf{TrapEval}(\mathsf{td}, F', X)$ takes as input the trapdoor $\mathsf{td}$, a function description $F'$, and $X \in \mathcal{X}$. It returns $\mathbf{R}_X \in \mathbb{Z}_q^{m\times m}$ and $\mathbf{H}_X \in \mathbb{Z}_q^{n\times n}$.

**Correctness:** For all $(F', \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$, $X \in \mathcal{X}$, and $(\mathbf{R}_X, \mathbf{H}_X) = \mathsf{TrapEval}(\mathsf{td}, F', X)$:

- Let $\mathsf{GL}(\mathbb{Z}_q, n)$ be the set of invertible matrices in $\mathbb{Z}_q^{n\times n}$. We require that $\mathbf{H}_X \in \mathsf{GL}(\mathbb{Z}_q, n) \cup \{\mathbf{0}\}$ and $\|\mathbf{R}_X\|_\infty \leq \beta$ hold with overwhelming probability over the trapdoor $\mathsf{td}$ corresponding to $F'$.
- We have

$$\mathsf{HEval}(F', X) = \mathbf{A}\mathbf{R}_X + \mathbf{H}_X\mathbf{G} \tag{5.2}$$

  where $\mathbf{G}$ is the gadget matrix of dimension $n$ from Lemma 13.

**Statistically close keys:** For $(F', \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $F \xleftarrow{\$} \mathsf{HGen}(1^\lambda)$, the statistical distance of $(\mathbf{A}, F)$ and $(\mathbf{A}, F')$ is at most $\gamma$.

**Well-distributed hidden matrices:** Consider the following experiment. We run $(F', \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and then

$$\mathcal{X} = (X_1, \ldots, X_Q, X^*) \xleftarrow{\$} \mathcal{A}(F')$$

for some algorithm $\mathcal{A}$. Further, let $(\mathbf{R}_{X^*}, \mathbf{H}_{X^*}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F', X^*)$ and $(\mathbf{R}_{X_i}, \mathbf{H}_{X_i}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F', X_i)$ for all $1 \leq i \leq Q$. We then require that

the events

$$\mathsf{coll}_{\mathsf{phf}} := \exists\, X \neq X' \in \mathcal{X} : \mathbf{H}_X = \mathbf{H}_{X'}, \qquad \mathsf{badChal}_{\mathsf{phf}} := \mathbf{H}_{X^*} \neq \mathbf{0}$$

are independent and that it holds that

$$(\varepsilon - \Pr[\mathsf{coll}_{\mathsf{phf}}]) \cdot \Pr[\neg\mathsf{badChal}_{\mathsf{phf}}] \geq \delta(t, \varepsilon) \tag{5.3}$$

for all algorithms $\mathcal{A}$ running in time $t$.

Note that in comparison to prior definitions of (lattice-based) programmable hash functions [ZCZ16, HK12], which considered "statistical" well-distribution that holds even for unbounded $\mathcal{A}$, we consider a "computational" variant. Following the approach from the previous chapter, the Equation (5.3) essentially captures the "balancedness" and allows us to apply balanced PHFs in partitioning proofs based on decisional assumptions without requiring an artificial abort.

## 5.3 Balanced Programmable Hash Functions from Blockwise Partitioning

In this section we describe how blockwise partitioning can be used in the construction of a balanced PHF. Recall that, for blockwise partitioning, we considered a family of hash functions $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ and assumed for simplicity that $n = \sum_{i=0}^{\ell} 2^i$ holds for some $\ell \in \mathbb{N}$. Note that this can be generalized to arbitrary $n$ but would only make the notation rather cumbersome without providing additional insight. For $H \in \mathcal{H}$, we then let $H_i : \{0,1\}^* \to \{0,1\}^{2^i}$ for all $i \in [\ell]_0$ such that

$$H(X) = H_0(X) \parallel \cdots \parallel H_\ell(X)$$

holds for all $X \in \{0,1\}^*$.

Encoding identities as full rank difference matrices. As preparation for the construction of our balanced PHF from blockwise partitioning, we discuss how a hash function output $H_i(X)$ is encoded as a matrix using the *full rank difference encoding function* (FRD) by Agrawal *et al.* [ABB10a] and adapt it to blockwise partitioning. Essentially, it guarantees that we can map bit strings injectively to matrices, such that the resulting matrices and their differences are invertible.

Informally, for a binary string $a \in \{0,1\}^{2^i}$, meaning $a$ is a potential output of $H_i$, we pad $a$ with zeros to be of length $n$ by first padding it with $\sum_{j=0}^{i-1} 2^j$ zeros in the front and with $\sum_{j=i+1}^{\ell} 2^j$ zeros in the end. We then canonically interpret it as a vector in $\mathbb{Z}_q^n$ and encode it with the full-rank difference encoding of [ABB10a]. We formalize the full rank difference encoding function and this process in the following definitions.

**Definition 32.** Let $f(\mathsf{Z})$ be an irreducible polynomial of degree $n$ in $\mathbb{Z}_q^n[\mathsf{Z}]$ and for $\mathbf{a} \in \mathbb{Z}_q^n$, let $g_a(\mathsf{Z}) \coloneqq \sum_{k=0}^{n-1} a_{k+1}\mathsf{Z}^k \in \mathbb{Z}_q^n[\mathsf{Z}]$. Then the function $\mathsf{FRD}(\mathbf{a}) : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ from [ABB10a] is defined as

$$\mathsf{FRD}(\mathbf{a}) \coloneqq \begin{pmatrix} \mathsf{coeffs}(g_{\mathbf{a}} \bmod f) \\ \mathsf{coeffs}(\mathsf{Z} \cdot g_{\mathbf{a}} \bmod f) \\ \mathsf{coeffs}(\mathsf{Z}^2 \cdot g_{\mathbf{a}} \bmod f) \\ \vdots \\ \mathsf{coeffs}(\mathsf{Z}^{n-1} \cdot g_{\mathbf{a}} \bmod f) \end{pmatrix} \in \mathbb{Z}_q^{n\times n},$$

where $\mathsf{coeffs}$ denotes the coefficients of a polynomial in $\mathbb{Z}_q^n[\mathsf{Z}]$.

Agrawal *et al.* [ABB10a] prove the following properties of $\mathsf{FRD}$ will be useful to us.

**Lemma 16** (Section 5 in [ABB10a])**.** *Let $\mathsf{FRD} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ be as defined in Definition 32, then the following holds:*

1. $\mathsf{FRD}$ *is injective.*
2. *There is an additive group $\mathbb{G} \subset \mathbb{Z}_q^{n\times n}$ such that each $\mathbf{H} \in \mathbb{G} \setminus \{\mathbf{0}\}$ is invertible and the range of $\mathsf{FRD}$ is a subset of $\mathbb{G}$*

We refer to [ABB10a, Section 5] for the proofs of the properties stated in Corollary 4. Given $\mathsf{FRD}$ and its properties, we now define the function $\mathsf{FRD}_i^{\mathsf{blk}}$, which formalizes the process of padding the outputs of $H_i$ that we described above.

**Definition 33.** For all $0 \le i \le \ell$ we define $\mathsf{FRD}_i^{\mathsf{blk}} : \{0,1\}^{2^i} \to \mathbb{Z}_q^{n\times n}$ to be the function that behaves as follows.

1. For an input $\mathbf{a} = (a_1, \ldots, a_{2^i}) \in \{0,1\}^{2^i}$, $\mathsf{FRD}_i^{\mathsf{blk}}$ lets $\mathsf{offset}_i \coloneqq \sum_{j=0}^{i-1} 2^j$ and sets $\mathbf{b}^T \coloneqq (b_1, \ldots, b_n) \in \mathbb{Z}_q^n$, where

$$b_k \coloneqq \begin{cases} a_{k-\mathsf{offset}_i} & \text{if } \mathsf{offset}_i < k \le \mathsf{offset}_i + 2^i \\ 0 & \text{otherwise} \end{cases}$$

   for all $1 \le k \le n$.

2. It then outputs $\mathsf{FRD}_i^{\mathsf{blk}}(\mathbf{a}) \coloneqq \mathsf{FRD}(\mathbf{b})$.

As we show below, the properties of $\mathsf{FRD}$ we stated in Lemma 16 carry over to $\mathsf{FRD}_i^{\mathsf{blk}}$.

**Corollary 2.** *Let $\mathsf{FRD}_i^{\mathsf{blk}} : \{0,1\}^{2^i} \to \mathbb{Z}_q^{n\times n}$ be as defined in Definition 33, then the following holds:*

1. $\mathsf{FRD}_i^{\mathsf{blk}}$ *is injective.*

2. *There is an additive group* $\mathbb{G} \subset \mathbb{Z}_q^{n \times n}$ *such that each* $\mathbf{H} \in \mathbb{G} \setminus \{0\}$ *is invertible and the range of* $\mathsf{FRD}_i^{\mathsf{blk}}$ *is a subset of* $\mathbb{G}$ *for all* $0 \leq i \leq \ell$.

*Proof.* The first claim, that $\mathsf{FRD}_i^{\mathsf{blk}}$ is injective for all $i \in [\ell]_0$, immediately follows from the fact that $\mathsf{FRD}$ is injective. This is because $\mathsf{FRD}_i^{\mathsf{blk}}$ only pads its input $\mathbf{a} \in \{0,1\}^{2^i}$ with zeros, resulting in $\mathbf{b} \in \mathbb{Z}_q^n$, and then outputs $\mathsf{FRD}(\mathbf{b})$. Since this padding process is injective, the injectivity of $\mathsf{FRD}$ immediately implies the injectivity of $\mathsf{FRD}_i^{\mathsf{blk}}$. Similarly, the second claim holds because the range of $\mathsf{FRD}_i^{\mathsf{blk}}$ is a subset of the range of $\mathsf{FRD}$ for all $i \in [\ell]_0$. □

Recall from Section 3.4.1, that for blockwise partitioning, we defined the algorithm $\mathsf{BPSmp}$ that on input $1^\lambda$, $t \in \mathbb{N}$ and $\varepsilon \in (0,1]$ with $t/\varepsilon < 2^\lambda$ outputs $(\mathsf{K}_0, \ldots, \mathsf{K}_\ell)$, where either $\mathsf{K}_i = \perp$ or $\mathsf{K} \in \{0,1\}^{2^i}$ holds for all $i \in [\ell]_0$. Essentially, we then showed in Lemma 6 that with a non-negligible probability

$$H_i(X) = \mathsf{K}_i \qquad \text{for all } i \in [\ell]_0 \text{ such that } \mathsf{K}_i \neq \perp$$

holds only for the challenge input $X = X^*$ and in particular for no query made by the adversary. This just enables a partitioning proof strategy. Our definition of $\mathsf{FRD}_i^{\mathsf{blk}}$ serves the purpose of expressing this line of thought in lattices. We concretise this in the following lemma.

**Lemma 17.** *Let* $\mathsf{BPSmp}$ *be as defined in Section 3.4.1 and let* $t \in \mathbb{N}, \varepsilon \in (0,1]$ *with* $t/\varepsilon < 2^\lambda$. *Then for* $(\mathsf{K}_0, \ldots, \mathsf{K}_\ell) \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t, \varepsilon)$, $\mathcal{I} = \{i : \mathsf{K}_i \neq \perp\} \subseteq [\ell]_0$ *and* $X \in \{0,1\}^*$ *it holds that*

$$-\left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\right) + \left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(H_i(X))\right) = 0 \iff \mathsf{K}_i = H_i(X) \text{ for all } i \in \mathcal{I}.$$

*Proof.* First, we observe that if $H_i(X) = \mathsf{K}_i$ for all $i \in \mathcal{I}$, then it holds that

$$\begin{aligned} &-\left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\right) + \left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(H_i(X))\right) \\ =&-\left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\right) + \left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\right) = 0, \end{aligned}$$

which proves the first direction of the equivalence. We prove the second direction by contradiction. Informally, we do so by observing that the first row of $\mathbf{H}_X$ consists of the differences between $\mathsf{K}_i$ and $H_i(X)$ over $\mathbb{Z}_q$ for all $i \in \mathcal{I}$. Hence, $\mathbf{H} = 0$ implies $H_i(X) = \mathsf{K}_i$ for all $i \in \mathcal{I}$, which contradiction the original assumption. We proceed by formalizing this proof by contradiction as follows.

Assume that there exists an index $i^* \in \mathcal{I}$ such that $H_{i^*}(X) \neq \mathsf{K}_i$ holds and $\mathbf{H}_X \coloneqq -\left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\right) + \left(\sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(H_i(X))\right) = 0$ also holds at the same time. Now for all $i \in \mathcal{I}$ and $1 \leq j \leq n$ we denote the $j$-th element of the first row of $\mathsf{FRD}_i^{\mathsf{blk}}(H_i(X))$ by $b_{i,j} \in \mathbb{Z}_q$. Analogously, we denote the $j$-th element of the first row of $\mathbf{H}_X$ by $h_j \in \mathbb{Z}_q$. We now make the following observations that follow immediately from the definition of $\mathsf{FRD}_i^{\mathsf{blk}}$ in Definition 33.

1. For all $i \in \mathcal{I}$ and for all $j \in \{1, \ldots, n\} \setminus \{\mathsf{offset}_i + 1, \ldots, \mathsf{offset}_i + 2^i\}$ we have $b_{i,j} = 0$.

2. It holds $(b_{i,\mathsf{offset}_i+1}, \ldots, b_{i,\mathsf{offset}_i+2^i}) = H_i(X)$ and $(h_{\mathsf{offset}_i+1}, \ldots, h_{\mathsf{offset}_i+2^i}) = \mathsf{K}_i$ for all $i \in \mathcal{I}$, where $\mathsf{offset}_i$ is as defined in Definition 33.

Combining the two observations yields that for all $i \in \mathcal{I}$ and $\mathsf{offset}_i + 1 \leq j \leq \mathsf{offset}_i + 2^i$ it holds that $[h_{\mathsf{offset}_i+1}, \ldots, h_{\mathsf{offset}_i+2^i}]^T = H_i(X) - \mathsf{K}_i$, where we interpret $H_i(X)$ and $\mathsf{K}_i$ as vectors in $\mathbb{Z}_q^{2^i}$. Recall that by our assumption above, it holds that $\mathbf{H}_X = 0$ and hence $H_{i^*}(X) - \mathsf{K}_{i^*} = 0$, if interpreted as vectors in $\mathbb{Z}_q^{2^i}$. This however contradicts $H_{i^*}(X) \neq \mathsf{K}_i$, which proves the lemma. □

Next, we show how we can use blockwise partitioning to construct balanced programmable hash function based and blockwise partitioning and then proof that it is indeed a balanced PHF.

**Construction 4.** Let $\mathcal{H} = \{\{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions with $n = 2\lambda + 3$ and let $\ell \coloneqq \lfloor \log(2\lambda + 3) \rfloor$. We define a $\mathcal{F}_{\mathsf{BLK}} = (\mathsf{HGen}_{\mathsf{blk}}, \mathsf{HEval}_{\mathsf{blk}})$ as follows.

$\mathsf{HGen}_{\mathsf{blk}}(1^\lambda)$ runs $H \xleftarrow{\$} \mathcal{H}$ and samples $\mathbf{B}, \mathbf{B}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and outputs the function description $F_{\mathsf{blk}} \coloneqq (H, \mathbf{B}, (\mathbf{B}_i)_{i \in [\ell]_0})$.

$\mathsf{HEval}_{\mathsf{blk}}(F_{\mathsf{blk}}, \mathsf{id})$ does the following:

1. It computes $\mathbf{H}_i \coloneqq \mathsf{FRD}_i^{\mathsf{blk}}(H_i(\mathsf{id}))$ for all $i \in [\ell]_0$.
2. It then computes $\mathbf{B}'_i \coloneqq \mathbf{B}_i \cdot \mathbf{G}^{-1}(\mathbf{H}_i \mathbf{G})$ for all $i \in [\ell]_0$.
3. It then returns
$$\mathbf{B}_{\mathsf{id}} \coloneqq \mathbf{B} + \sum_{i=0}^{\ell} \mathbf{B}'_i.$$

**Lemma 18.** *If $\mathcal{H}$ is instantiated with a family of weak near-collision resistant hash functions, then $\mathcal{F}_{\mathsf{BLK}}$ above is a $(\beta, \gamma, \delta)$-balanced programmable hash function for $\beta \coloneqq 1 + (\ell + 1) \cdot m$, $\gamma = \mathsf{negl}(\lambda)$ and $\delta(t, \varepsilon) \coloneqq \varepsilon^2/(32t^2 - 16t)$, provided that $\mathsf{TrapGen}_{\mathsf{blk}}$ is only run on inputs $t \in \mathbb{N}$, $\varepsilon \in [0,1]$ such that $t/\varepsilon < 2^\lambda$.*

*Proof.* We begin by describing the algorithms $\mathsf{TrapGen}_{\mathsf{blk}}$ and $\mathsf{TrapEval}_{\mathsf{blk}}$.

$\mathsf{TrapGen}_{\mathsf{blk}}(1^\lambda, t, \varepsilon, \mathbf{A})$ works as follows:

1. It samples $H \xleftarrow{\$} \mathcal{H}$
2. It runs $\mathsf{K} \xleftarrow{\$} \mathsf{BPSmp}(1^\lambda, t, \varepsilon)$, parses $\mathsf{K} = (\mathsf{K}_0, \ldots, \mathsf{K}_\ell)$ and sets $\mathcal{I} \coloneqq \{i \in [\ell]_0 : \mathsf{K}_i \neq \bot\}$.
3. It then samples $\mathbf{R}, \mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$ for all $0 \leq i \leq \ell$

4. It sets
$$\mathbf{B}_i \coloneqq \begin{cases} \mathbf{A}\mathbf{R}_i + \mathbf{G} & \text{if } \mathsf{K}_i \neq \bot \\ \mathbf{A}\mathbf{R}_i & \text{if } \mathsf{K}_i = \bot \end{cases}$$
for all $i \in [\ell]_0$.
5. It computes
$$\mathbf{B} \coloneqq \mathbf{A}\mathbf{R} - \left( \sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\mathbf{G} \right).$$
6. It returns $F_{\mathsf{blk}} \coloneqq (H, \mathbf{B}, (\mathbf{B}_i)_{i \in [\ell]_0})$ and $\mathsf{td} \coloneqq (\mathbf{R}, (\mathbf{R}_i)_{i \in [\ell]_0})$.

$\mathsf{TrapEval}_{\mathsf{blk}}(\mathsf{td}, F_{\mathsf{blk}}, \mathsf{id})$ works as follows:

1. It computes $\mathbf{H}_i \coloneqq \mathsf{FRD}_i^{\mathsf{blk}}(H_i(\mathsf{id}))$ for all $i \in [\ell]_0$.
2. It sets $\mathbf{R}'_i \coloneqq \mathbf{R}_i \mathbf{G}^{-1}(\mathbf{H}_i \mathbf{G})$.
3. It outputs
$$\mathbf{R}_{\mathsf{id}} \coloneqq \mathbf{R} + \sum_{i=0}^{\ell} \mathbf{R}'_i \qquad \text{and} \qquad \mathbf{H}_{\mathsf{id}} \coloneqq \sum_{i \in \mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(H_i(\mathsf{id}))\mathbf{G}.$$

We first notice that $\mathsf{TrapGen}_{\mathsf{blk}}$ and $\mathsf{TrapEval}_{\mathsf{blk}}$ are syntactically correct and can be executed in probabilistic polynomial time. To show correctness, we need to prove that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$ and that $\mathsf{HEval}_{\mathsf{blk}}(F_{\mathsf{AS}}, \mathsf{id}) = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$ for $(F_{\mathsf{blk}}, \mathsf{td}) = \mathsf{TrapGen}_{\mathsf{blk}}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}}) = \mathsf{TrapEval}_{\mathsf{blk}}(\mathsf{td}, F_{\mathsf{AS}}, \mathsf{id})$. We start by proving $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$.

Proving $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$. First, observe that $\|\mathbf{R}'_i\|_\infty = \|\mathbf{R}_i \mathbf{G}^{-1}(\mathbf{H}_i G)\|_\infty \leq m$ holds since $\mathbf{R}_i, \mathbf{G}^{-1}(\mathbf{H}_i \mathbf{G}) \in \{-1, 1\}^{m \times m}$ and therefore their product $\mathbf{R}'_i \in \mathbb{Z}_q^{m \times m}$ can not contain any element of absolute value larger than $m$. We then have that

$$\|\mathbf{R}_{\mathsf{id}}\|_\infty = \left\| \mathbf{R} + \sum_{i=0}^{\ell} \mathbf{R}'_i \right\|_\infty \leq \|\mathbf{R}\|_\infty + \sum_{i=0}^{\ell} \|\mathbf{R}'_i\|_\infty \leq 1 + (\ell + 1)m = \beta$$

holds, where the last inequality follows from $\mathbf{R} \in \{-1, 1\}^{m \times m}$ and $\|\mathbf{R}'_i\|_\infty \leq m$.

Further, the trapdoor keys are statistically close because it holds that $(\mathbf{A}, \mathbf{B}_K)$ and $(\mathbf{A}, \mathbf{B}_1)$ have only a negligible statistical difference by the Leftover-Hash Lemma (Lemma 15).

Proving $\mathbf{B}_{\mathsf{id}} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$. To complete the proof of correctness, we show that Equation (5.2) from the definition of balanced PHFs holds. That is, we show that it holds that

$$\mathsf{HEval}_{\mathsf{blk}}(F'_{\mathsf{blk}}, \mathsf{id}) = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}.$$

We observe that

$$\begin{aligned}\mathsf{HEval}_{\mathsf{blk}}(F'_{\mathsf{blk}}, \mathsf{id}) &= \mathbf{B} + \sum_{i=0}^{\ell} \mathbf{B}'_i = \mathbf{B} + \sum_{i=0}^{\ell} \mathbf{B}_i \cdot \mathbf{G}^{-1}(\mathbf{H}_i \mathbf{G}) \qquad (5.4)\\ &= \mathbf{A}\mathbf{R} - \left(\sum_{i\in\mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\mathbf{G}\right) + \left(\sum_{i\in\mathcal{I}} \mathbf{A}\mathbf{R}'_i + x_i \mathbf{H}_i \mathbf{G}\right)\\ &= \mathbf{A}\left(\mathbf{R} + \sum_{i=0}^{\ell} \mathbf{R}'_i\right) - \left(\sum_{i\in\mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(\mathsf{K}_i)\mathbf{G}\right) + \left(\sum_{i\in\mathcal{I}} \mathsf{FRD}_i^{\mathsf{blk}}(H_i(\mathsf{id}))\mathbf{G}\right)\\ &= \mathbf{A}\mathbf{R}_{\mathsf{id}} - \mathbf{H}_{\mathsf{id}}\mathbf{G}\end{aligned}$$

holds, where Equation (5.4) holds by the definition of $\mathsf{HEval}_{\mathsf{blk}}(F'_{\mathsf{blk}}, \mathsf{id})$ and all further equations follow from the definitions of $\mathsf{TrapGen}_{\mathsf{blk}}(1^\lambda, \mathbf{A})$ and the trapdoor evaluation algorithm $\mathsf{TrapEval}_{\mathsf{blk}}(\mathsf{td}, F'_{\mathsf{blk}}, \mathsf{id})$. Furthermore, notice that $\mathbf{H}_{\mathsf{id}} \in \mathsf{GL}(\mathbb{Z}_q, n)$ because it is a sum of the outputs of $\mathsf{FRD}_i^{\mathsf{blk}}$ (for different $i$), which all produce outputs in an additive subgroup of $\mathsf{GL}(\mathbb{Z}_q, n)$ by Lemma 17. This complete the proof of correctness.

Well-distributedness. Let $\mathcal{A}$ be an algorithm that outputs $\mathcal{Q}^* \coloneqq \{\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}, \mathsf{id}^*\}$ on input $F_{\mathsf{blk}}$ with $(F_{\mathsf{blk}}, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$. Furthermore, let

$$\begin{aligned}(\mathbf{R}_{\mathsf{id}^*}, \mathbf{H}_{\mathsf{id}^*}) &\coloneqq \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{blk}}, \mathsf{id}^*)\\ (\mathbf{R}_{\mathsf{id}^{(i)}}, \mathbf{H}_{\mathsf{id}^{(i)}}) &\coloneqq \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{blk}}, \mathsf{id}^{(i)}) \text{ for } 1 \le i \le Q\end{aligned}$$

and let $\mathsf{id}^{(Q+1)} \coloneqq \mathsf{id}^*$ and define the events $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}$ as in Definition 31 as

$$\mathsf{coll}_{\mathsf{phf}} \coloneqq \exists\, \mathsf{id} \neq \mathsf{id}' \in \mathcal{Q}^* : H_{\mathsf{id}} = H_{\mathsf{id}'}, \qquad \mathsf{badChal}_{\mathsf{phf}} \coloneqq H_{\mathsf{id}^*} \neq \mathbf{0}.$$

First, recall that we have $\mathbf{H}_{\mathsf{id}} = \mathbf{H}'_{\mathsf{id}}$ *if and only if* $H_i(\mathsf{id}) = H_i(\mathsf{id}')$ for all $i \in \mathcal{I}$ by Lemma 17. We thus have that

$$\mathsf{coll} \iff \mathsf{coll}_{\mathsf{phf}} \qquad \text{and} \qquad \mathsf{badChal} \iff \mathsf{badChal}_{\mathsf{phf}}.$$

Thus, $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}_{\mathsf{phf}}$ are independent because $\mathsf{coll}$ and $\mathsf{badChal}$ are independent by Lemma 6. Also by Lemma 6, we thus have that

$$(\varepsilon - \Pr[\mathsf{coll}_{\mathsf{phf}}]) \cdot \Pr[\neg\mathsf{badChal}_{\mathsf{phf}}] \ge \frac{\varepsilon^2}{32t^2 - 16t}$$

holds, which is non-negligible if $t$ is polynomial and $\varepsilon$ is non-negligible in $\lambda$. □

## 5.4 Hash Functions with Exponential Collision Resistance

In the previous section just as in the previous chapter, we just assumed the plausible but nonetheless unproven existence of families of TCR and wNCR hash functions. In this section, we address this shortcoming. To that end, we introduce *exponentially-collision resistant* (ECR) hash functions, which essentially requires that any algorithm for finding collision takes exponential time in the *output length* of the hash function. Thus, while the notion of ECR hash functions is similar in spirit, it is a weaker notion than TCR and wNCR for hash functions. We show how to explicitly construct ECR hash functions based on lattice assumptions. To this end we define the *exponential short integer solution* (eSIS) assumption, which essentially extends the standard SIS assumption [Ajt96] with a concrete bound on the success probability of any adversary running within a certain amount of time. We stress that this bound holds for all currently known algorithms for SIS and closely related computation problems [BGLS19, APS15, BKW00, ACD+18, Duc18, MW16, ADH+19, SE94, BDGL16, Alb17, AGVW17] and also matches how one would instantiate SIS parameters in practice. We also discuss the relation of eSIS to worst-case lattice problems $\mathrm{SIVP}_\gamma$ and GapSVP with polynomial approximation factors. The main idea of our construction is to follow the well-known construction of collision resistant hash functions from SIS by [Ajt96] to construct suitable *compression functions*, which can then be used in the Merkle-Dåmgard transformation to obtain ECR hash function families.

Since we will instantiate ECR hash functions from a concrete lattice hardness assumption, we will consider families of hash functions of the form

$$\mathcal{H}_n = \{H : \{0,1\}^\lambda \to \mathbb{Z}_q^n\},$$

where $n \in \mathbb{N}$ and $q = q(n) \in \mathbb{N}$ is a prime whose size may depend on $n$.

**Definition 34** (Exponential Collision Resistance)**.** We say that a family of hash functions $\mathcal{H}_n$ is $(c,d)$-*exponentially-collision resistant* (ECR) for constants $c \geq 1, d > 0$ if for every probabilistic algorithm $\mathcal{A}$ running in time at most $t \in \mathbb{N}$ it holds that

$$\Pr_{H \xleftarrow{\$} \mathcal{H}_n}[H(x) = H(y) \mid (x,y) \xleftarrow{\$} \mathcal{A}(H)] \leq \frac{t^c}{2^{n \cdot d}}.$$

*Remark* 10. We observe that every family wNCR or TCR hash functions is also a family of ECR hash functions for $q := 2$, $c := 2$ and $d := 1$ by setting $\eta := n$.

We will show how to construct ECR hash function families that satisfy this bound for *every* $n \in \mathbb{N}$, even for small values which are too small to achieve standard collision resistance. The construction is based on a *concrete* formulation of the SIS problem. Furthermore, we sometimes just say that a family of hash functions is ECR if the constants $c$ and $d$ are irrelevant or clear from the respective context.

Technical lemma. The following lemma will be useful to apply exponential collision resistance in our construction of programmable hash functions and forms, together with Lemma 20, the equivalent to Lemma 4 for ECR hash functions. It essentially asserts that we can choose $n = 2^{i^*} = \mathcal{O}(\lambda)$, for some carefully chosen $i^* \in \mathbb{N}$, such that no adversary running in time $t$ can break the ECR of the hash function with probability better than $\varepsilon/2$ for some $\varepsilon \in (0, 1]$.

**Lemma 19.** *Let $c \geq 1$ and $d > 0$ be constants, and let $t \in \mathbb{N}$ and $\varepsilon \in (0, 1]$ with $t/\varepsilon < 2^\lambda$. Then for $i^* \coloneqq \lceil \log((\log(t^c/\varepsilon) + c + 1)/d) \rceil$ and $n \coloneqq 2^{i^*}$ it holds that:*

$$i^* \leq \left\lceil \log\left(\frac{c(\lambda+1)+1}{d}\right)\right\rceil \qquad \text{and} \qquad \frac{\varepsilon}{2} \geq \frac{t^c}{2^{d\cdot n - c}}.$$

*Proof.* We prove the first inequality as follows.

$$\begin{aligned} i^* = \left\lceil \log\left(\frac{\log(t^c/\varepsilon)+c+1}{d}\right)\right\rceil &\leq \left\lceil \log\left(\frac{\log(t^c/\varepsilon^c)+c+1}{d}\right)\right\rceil \\ &\leq \left\lceil \log\left(\frac{c\log(t/\varepsilon)+c+1}{d}\right)\right\rceil \\ &\leq \left\lceil \log\left(\frac{c\lambda+c+1}{d}\right)\right\rceil = \left\lceil \log\left(\frac{c(\lambda+1)+1}{d}\right)\right\rceil \end{aligned}$$

To show the second inequality, we observe that $n = 2^{\lceil \log((\log(t^c/\varepsilon)+c+1)/d)\rceil}$ implies that $n \geq \frac{\log\left(\frac{t^c}{\varepsilon}\right)+c+1}{d}$. Then the proof is straightforward:

$$\frac{t^c}{2^{d\cdot n - c}} \leq \frac{t^c}{2^{d\frac{\log\left(\frac{t^c}{\varepsilon}\right)+c+1}{d} - c}} = \frac{t^c}{2^{\log\left(\frac{t^c}{\varepsilon}\right)+1}} = \frac{t^c}{2\frac{t^c}{\varepsilon}} = \frac{\varepsilon}{2}$$

□

### 5.4.1 Concrete Exponential Hardness of SIS

The *exponential short integer solution* (eSIS) assumption essentially extends the standard *short integer solution* (SIS) assumption, as considered in [Ajt96], with a concrete exponential bound on the success probability of any adversary running within a certain amount of time. We stress that this bound holds for all currently known algorithms [BGLS19, APS15, BKW00, ACD+18, Duc18, MW16, ADH+19, SE94, BDGL16, Alb17, AGVW17] and also matches a reasonable choice of parameters for SIS when instantiated in practice. A sub-exponential time algorithm breaking SIS, and thus the eSIS assumption, would be a major algorithmic and cryptanalytic breakthrough.

**Definition 35** (Exponential SIS (eSIS) assumption)**.** There are constants $c \geq 1$ and $d > 0$ and polynomials $m, q : \mathbb{N} \to \mathbb{N}$ such that for all $n \in \mathbb{N}$ and all algorithms $\mathcal{A}$ we have that

$$\Pr\left[\mathbf{A}\mathbf{x} = 0 \bmod q(n) \wedge \mathbf{x} \in \{-1, 0, 1\}^{m(q)} \wedge \mathbf{x} \neq \mathbf{0} \mid \mathbf{x} = \mathcal{A}(\mathbf{A})\right] \leq \frac{t^c}{2^{d\cdot n}},$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m(n)}$ and $t \in \mathbb{N}$ is the maximal runtime of $\mathcal{A}$ on inputs from $\mathbb{Z}_q^{n \times m(n)}$.

Observe that Definition 35 imposes a particularly strict limit on the norm of solution vectors, by requiring $\mathbf{x} \in \{-1, 0, 1\}^{m(n)}$, which makes the assumption weaker than a more general bound $\gamma$. Moreover, we will just write $m$ instead of $m(n)$ and $q$ instead of $q(n)$ if it is clear from context.

Applicability of worst-case reductions to eSIS. Micciancio and Peikert [MP13, Theorem 1] show that for $m := n \cdot \lceil 3\log(n) + 4 \rceil + 1$ and $q$ the smallest prime larger than $4n^3$ any polynomial time algorithm breaking the SIS assumption implies a polynomial time algorithm for worst case $\mathrm{SIVP}_\gamma$ with a polynomial approximation factor $\gamma$.[1] It seems that this result can also be adopted to showing that an algorithm breaking the eSIS assumption implies a sub-exponential time algorithm for worst case $\mathrm{SIVP}_\gamma$ with a polynomial approximation factor $\gamma$ (which would be a breakthrough result for worst-case lattice problems). We did not prove this formally, though.

Furthermore, the eSIS assumption is also supported by the *Exponential Time Hypothesis* (ETH) for GapSVP by Lombardi and Vaikuntanathan [LV20, Conjecture 2.1] by applying Micciancio's and Regev's reduction [MR07, Theorem 5.23]. The option to base the eSIS assumption on the ETH for GapSVP with polynomial approximation factors is worth noting, because it provides a fallback in case that $\mathrm{SIVP}_\gamma$ with polynomial approximation factors $\gamma$ surprisingly turns out to be solvable in sub-exponential time.

We want to thank Martin Albrecht and Damien Stehlé here for helpful discussions and further information supporting the plausibility of the eSIS assumption.

## 5.4.2 Constructing ECR Hash Functions from eSIS

We now use the eSIS assumption to construct a family of ECR hash functions. The analysis of our construction will require a concrete instantiation of SIS parameters. As discussed in Section 5.4.1, in this chapter we follow [MP13, Theorem 1], which reduces worst-case lattice problems to SIS for certain parameters. Therefore, for the remained of this section, let $q$ be the smallest prime larger than $4n^3$ and set $m := n \cdot \lceil 3\log(n) + 4 \rceil + 1$.

Collision resistant compression function. We first show that $f_{\mathbf{A}}(\mathbf{x}) := \mathbf{A}\mathbf{x}$ for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{x} \in \{0,1\}^m$ is an ECR *compression* function. Standard collision resistance for sufficiently large $n$ was shown by Ajtai [Ajt96]. We show a *concrete* exponential bound with respect to the eSIS assumption, which yields ECR even for small $n$.

**Theorem 10.** *Let* $f_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n$, $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$. *Then:*

[1] Taking the variables from [MP13, Theorem 1], we implicitly set $\beta = \sqrt{m}, \beta_\infty = 1$ and $\delta = 1$. It can then easily be verified that $q = 4n^3 \geq \beta \cdot n^\delta$ and $q > \beta$ for all $n \in \mathbb{N}$.

1. *$f_{\mathbf{A}}$ is a compression function, since $m > n \cdot (\lfloor \log(q) \rfloor + 1)$.*
2. *$f_{\mathbf{A}}$ is exponentially-collision resistant if eSIS holds. Concretely, eSIS implies that there are constants $c \geq 1$ and $d > 0$ such that for all algorithm $\mathcal{A}$ running in time at most $t \in \mathbb{N}$ on inputs from $\mathbb{Z}_q^{n \times m}$ we have that*

$$\Pr\left[\mathbf{Ax} = \mathbf{Ay} \bmod q \wedge \mathbf{x}, \mathbf{y} \in \{0,1\}^m \wedge \mathbf{x} \neq \mathbf{y} \;\middle|\; \begin{matrix} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m} \\ (\mathbf{x}, \mathbf{y}) = \mathcal{A}(\mathbf{A}) \end{matrix}\right] \leq \frac{t^c}{2^{d \cdot n}}. \tag{5.5}$$

*Proof.* We first show that $m \geq n \cdot (\lfloor \log(q) \rfloor + 1)$. For this, observe that $q \leq 8n^3$ by the Bertrand-Chebyshev Theorem (see Theorem 9). It thus holds that

$$\begin{aligned} n \cdot (\lfloor \log(q) \rfloor + 1) &\leq n \cdot (\log(q) + 1) \leq n \cdot (3\log(n) + 4) < n \cdot \lceil 3\log(n) + 4 \rceil + 1 \\ &= m. \end{aligned}$$

We now show that Equation (5.5) holds if the eSIS assumption holds. This part closely follows the proof from [Ajt96]. Assume that there is an $n \in \mathbb{N}$ and an algorithm $\mathcal{A}$ running in time $t_{\mathcal{A}} \in \mathbb{N}$ on inputs from $\mathbb{Z}_q^{n \times m}$ for which there are no constants $c \geq 1$ and $d > 0$ such that Equation (5.5) holds. We then construct an algorithm $\mathcal{B}$ that, on input $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, runs $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{A}(\mathbf{A})$ and outputs $\mathbf{x} - \mathbf{y}$. If $\mathbf{Ax} = \mathbf{Ay}$, then we have $\mathbf{A}(\mathbf{x} - \mathbf{y}) = \mathbf{Ax} - \mathbf{Ay} = \mathbf{0}$. Furthermore, we have $\mathbf{0} \neq (\mathbf{x} - \mathbf{y}) \in \{-1, 0, 1\}^m$. Hence, $\mathcal{B}$ breaks the eSIS assumption. □

Collision resistant hashing. Now that we have proven that $f_{\mathbf{A}}$ is a collision resistant compression function, we can apply the Merkle-Dåmgard construction [Mer90, Dam90] to obtain a family of hash functions with arbitrary input lengths. We will construct a function with *fixed* input space $\{0,1\}^\lambda$, where $\lambda$ is the security parameter, since this avoids the need to encode the input length in the input to the Merkle-Dåmgard construction and is sufficient for our purposes. If required for a particular application, the input space can be generically extended to $\{0,1\}^*$ by first applying a standard collision resistant function $h : \{0,1\}^* \to \{0,1\}^\lambda$.

In order to construct a hash function family $\mathcal{H}_n$ for some $n \in \mathbb{N}$, we let $m := n \cdot \lceil 3\log(n) + 4 \rceil + 1$ and set $q$ to be the smallest prime larger than $4n^3$, as discussed above in Section 5.4.1. A random hash function $H \xleftarrow{\$} \mathcal{H}_n$ is sampled by first choosing a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$. The matrix $\mathbf{A}$ defines the above compression function $f_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n$, which in turn will define a hash function $H_{\mathbf{A}}$ as follows.

By encoding vectors in $\mathbb{Z}_q^n$ as binary vectors, we can view $f_{\mathbf{A}}$ as a function

$$f_{\mathbf{A}} : \{0,1\}^m \to \{0,1\}^\alpha$$

for $\alpha := n \cdot (\lfloor \log(q) \rfloor + 1)$. Since Theorem 10 shows that this is a compression function, we have $m > \alpha$. By setting $r := m - \alpha \geq 1$, we can apply the Merkle-Dåmgard transformation to process exactly $r$ input bits per call of the underlying compression function. To describe the function

$$H_{\mathbf{A}} : \{0,1\}^\lambda \to \mathbb{Z}_q^n$$

we will henceforth assume without loss of generality that $\lambda$ is an integer multiple of $r$. This is without loss of generality, because one can always pad constant-length inputs until their length is a multiple of $r$. Applying the Merkle-Dåmgard transformation [Mer90, Dam90] to $f_{\mathbf{A}}$ yields the following construction.

**Construction 5.** For any $n \in \mathbb{N}$, we define family $\mathcal{H}_n$ as follows. Let $m := n \cdot \lceil 3\log(n) + 4 \rceil + 1$ and let $q = q(n)$ be the function that returns the smallest prime larger than $4n^3$.

**Function sampling:** Choose $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and output $H := \mathbf{A}$.

**Function evaluation:** On input $H = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $x \in \{0,1\}^\lambda$:

1. Let $r := m - \alpha$ and split $x$ into $r$-bit chunks $B_i$ such that $x = B_1 \| \cdots \| B_L$, where $\|$ denotes concatenation.
2. Set $z_0 := 0^\alpha$ and for all $i \in [L]$ let $z_i := f_{\mathbf{A}}(B_i \parallel z_{i-1})$.
3. Output $f_{\mathbf{A}}(B_L \parallel z_{L-1})$.

The following corollary essentially applies the security proof of the Merkle-Dåmgard transform to the above construction. We prove it for completeness and to verify the claimed bound.

**Corollary 3.** *For any $n \in \mathbb{N}$, $\mathcal{H}_n$ from Construction 5 is a family of exponentially collision resistant hash functions in the sense of Definition 35 under the eSIS assumption.*

*Proof.* Suppose that $\mathcal{H}_n$ is not exponentially-collision resistant. That is, for all constants $c \geq 1$ and $d > 0$ there is an adversary $\mathcal{A}$ that runs in time $t_{\mathcal{A}}$ such that

$$\Pr\left[H_{\mathbf{A}}(x) = H_{\mathbf{A}}(y) : \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}; (x,y) \xleftarrow{\$} \mathcal{A}(\mathbf{A})\right] > \frac{t^c}{2^{d \cdot n}}$$

We then construct an adversary $\mathcal{B}$ that breaks the exponential collision resistance of the compression function $f_{\mathbf{A}}$. $\mathcal{B}$ receives as input $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and runs $(x,y) \xleftarrow{\$} \mathcal{A}(\mathbf{A})$ with $x, y \in \{0,1\}^\lambda$. Write

$$x = B_1^{(x)} \parallel \cdots \parallel B_L^{(x)} \qquad \text{and} \qquad y = B_1^{(y)} \parallel \cdots \parallel B_L^{(y)}.$$

$\mathcal{B}$ then sets $z_0^{(x)} := z_0^{(y)} := 0^\alpha$ and computes $z_i^{(x)} := f_{\mathbf{A}}(B_i^{(x)} \parallel z_{i-1}^{(x)})$ and $z_i^{(y)} := f_{\mathbf{A}}(B_i^{(y)} \parallel z_{i-1}^{(y)})$ for all $i \in [L]$. If $z_L^{(x)} \neq z_L^{(y)}$, then $\mathcal{B}$ aborts. Otherwise, it lets $i^* \in [L-1]$ be the largest index such that $z_{i^*}^{(x)} \neq z_{i^*}^{(y)}$ and then outputs $\mathbf{x} := (B_{i^*}^{(x)} \parallel z_{i^*}^{(x)}) - (B_{i^*}^{(y)} \parallel z_{i^*}^{(y)}) \in \{-1,0,1\}^m$.

Observe that $\mathbf{x} \neq \mathbf{0}$ because $z_{i^*}^{(x)} \neq z_{i^*}^{(y)}$. Furthermore, we have $z_{i^*+1}^{(x)} = z_{i^*+1}^{(y)}$ because $i^* \in [L-1]$ is the largest index such that $z_{i^*}^{(x)} \neq z_{i^*}^{(y)}$. Hence we obtain that

$$z_{i^*+1}^{(x)} = z_{i^*+1}^{(y)} \Leftrightarrow f_{\mathbf{A}}(B_{i^*}^{(x)} \parallel z_{i^*}^{(x)}) = f_{\mathbf{A}}(B_{i^*}^{(y)} \parallel z_{i^*}^{(y)}) \Leftrightarrow \mathbf{A}\mathbf{x} = \mathbf{0}.$$

□

Guessing the output of an ECR hash function. The following lemma provides a bound on the probability of "guessing" an $(c, d)$-ECR hash output, as a function of the size of $n$. Note that this is where the super-polynomial loss in our reduction originates from, because $q$ grows with $i^*$. If one constructed a provably secure ECR-hash function with constant $q$ or even over $\{0,1\}$, then this would allow us to avoid the quasi-polynomially bounded loss in our reductions, which however is even smaller than quasi-polynomial. Thus, if we instantiate an ECR hash function with an TCR or wNCR hash function as discussed in Remark 10, then we would indeed have $q = 2$. However, we leave the question of constructing ECR hash functions with a constant $q$ from well-studied computational problems as an open research problem (see Research Question 6).

**Lemma 20.** *Let $c \geq 1$ and $d > 0$ be constants, and let $t \in \mathbb{N}$ and $1/\varepsilon$ both be polynomial in $\lambda$. For*

$$i^* \coloneqq \left\lceil \log\left(\frac{\log(t^c/\varepsilon) + c + 1}{d}\right)\right\rceil,$$

*$n \coloneqq 2^{i^*}$ and $q$ the smallest prime larger than $4n^3$, we then have that $q^n = \lambda^{\mathcal{O}(\log\log\lambda)}$.*

*Proof.* Observe that since both $t$ and $1/\varepsilon$ are polynomial in $\lambda$, we have that there is a constant $f$ such that for $\lambda$ large enough it holds that

$$\frac{\log(t^c/\varepsilon) + c + 1}{d} \leq f\log(\lambda)$$

and hence we have that

$$n = 2^i \leq 2 \cdot 2^{\log\left(\frac{\log(t^c/\varepsilon)+1}{d}\right)} \leq 2f\log(\lambda)$$

for $\lambda$ large enough. Observe that by the Bertrand-Chebyshev Theorem (see Theorem 9) we have that $q \leq 8n^3$. We can therefore conclude the following for $\lambda$ large enough:

$$\begin{aligned} q^n &\leq \left(8n^3\right)^n \leq 8^{2\cdot f\log(\lambda)} \cdot n^{6\cdot f\log(\lambda)} \leq 2^{6\cdot f\log(\lambda)}\left(2f\log(\lambda)\right)^{6\cdot f\log(\lambda)} \\ &= (2\cdot 2\cdot f)^{6\cdot f\log(\lambda)} \cdot \log(\lambda)^{6\cdot f\log(\lambda)} = \lambda^{\mathcal{O}(1)} \cdot 2^{6\cdot f\log(\lambda)\cdot\log\log(\lambda)} \\ &= \lambda^{\mathcal{O}(1)} \cdot \lambda^{6\cdot f\cdot\log\log(\lambda)} = \lambda^{\mathcal{O}(\log\log(\lambda))} \end{aligned}$$

□

### 5.4.3 Instantiating Multiple ECR Hash Functions in Parallel

Our constructions of balanced PHFs will use $1 + \ell = \mathcal{O}(\log(\lambda))$ many copies of the above ECR hash function in parallel, with exponentially increasing range size. Therefore it will be convenient to define

$$n(i) \coloneqq 2^i, \qquad m(i) = n(i) \cdot \lceil 3\log(n(i)) + 4\rceil + 1, \qquad q(i) = q'(n(i)),$$

where $q'(n)$ is the function that returns the smallest prime larger than $4n^3$. Since the description of the hash functions has to be contained in the master public key of an IB-KEM or digital signature scheme and we are keen to keep the size of these as small as possible, we study how large the description of these $\ell$ many hash functions together are and show that they are asymptotically as large as a constant number matrices used in the context of constructions based on LWE or SIS.

**Lemma 21.** *Let $\ell = \mathcal{O}(\log(\lambda))$ and let $H_i \xleftarrow{\$} \mathcal{H}_{n(i)}$ for all $0 \leq i \leq \ell$. Then the hash functions $H_0, \ldots, H_\ell$ can all together be described by $\mathcal{O}(\lambda^2 \log(\lambda))$ many elements from $\{0, \ldots, q(\ell) - 1\}$.*

*Remark* 11. We count elements in $\{0, \ldots, q(\ell) - 1\}$ instead of $\mathbb{Z}_{q(i)}$ for the different moduli $q(i)$. We do so for simplicity because it allows us to disregard the different modulus sizes of the different hash functions. Note that we thus overestimate the size of the descriptions of the ECR hash functions.

*Proof.* First, recall that a hash function $H_i \in \mathcal{H}_i$ is represented by a matrix $\mathbf{A}_i \in \mathbb{Z}_{q(i)}^{n(i) \times m(i)}$ and is thus encoded by $g_i \coloneqq n(i) \cdot m(i)$ many elements from $\{0, \ldots, q(\ell) - 1\}$. We upper bound $g_i$ as follows.

$$g_i = 2^i \cdot 2^i \left\lceil 3\log(2^i) + 4 \right\rceil + 1 \leq 2^{2i} \cdot (3i + 5) + 1 = 3 \cdot i \cdot 4^i + 5 \cdot 4^i + 1$$

Thus, for any $\ell \in \mathbb{N}$, we have that

$$\begin{aligned}
\sum_{i=0}^{\ell} g_i &\leq 3 \cdot \left( \sum_{i=0}^{\ell} 4^i \cdot i \right) + 5 \cdot \left( \sum_{i=0}^{\ell} 4^i \right) + \ell + 1 \\
&= 3 \cdot \frac{\ell \cdot 4^{\ell+2} - (\ell+1) \cdot 4^{\ell+1} + 4}{3^2} + 5 \cdot \frac{4^{\ell+1} - 1}{3} + \ell + 1 \qquad (5.6) \\
&\leq \ell \cdot 4^{\ell+2} + 4 + 2 \cdot 4^{\ell+1} + \ell + 5 \\
&= \ell \cdot 2^{2\ell+4} + 2^{2\ell+3} + \ell + 5 \\
&= 16 \cdot \ell \cdot \left(2^\ell\right)^2 + 8 \cdot \left(2^\ell\right)^2 + \ell + 1 = \left(2^\ell\right)^2 (16 \cdot \ell + 8) + \ell + 5,
\end{aligned}$$

where Equation (5.6) follows from the closed-form formulas for the geometric series and a variant of it for all $t \in \mathbb{N}$ and $x \neq 1$:

$$\sum_{k=0}^{t} x^k \cdot k = \frac{k \cdot x^{n+2} - (k+1)x^{k+1} + x}{(x-1)^2} \qquad \sum_{k=0}^{t} x^k = \frac{x^{k+1} - 1}{x - 1}$$

Now let $f > 0$ such that $\ell \leq f \log(\lambda)$. Such a constant exists since $\ell = \mathcal{O}(\log(\lambda))$. We can then conclude the proof as follows.

$$\begin{aligned}
\sum_{i=0}^{\ell} g_i &\leq (2^\ell)^2 \cdot (16 \cdot \ell + 8) + \ell + 5 \\
&\leq 2^{2f\log(\lambda)} \cdot (16 \cdot f\log(\lambda) + 8) + f\log(\lambda) + 5 \\
&= \lambda^{2f} \cdot (16 \cdot f\log(\lambda) + 8) + f\log(\lambda) + 5 \\
&= \mathcal{O}(\lambda^2 \cdot \log(\lambda)) + \mathcal{O}(\lambda^2) + \mathcal{O}(\log(\lambda)) = \mathcal{O}(\lambda^2 \cdot \log(\lambda))
\end{aligned}$$

□

## 5.5 Constant-Size Balanced PHFs

This section will extend a technique by Alperin-Sheriff [Alp15], originally used to construct digital signatures with constant size keys from SIS. Previously, this technique achieved only *non-adaptive* security, and therefore could not be used to construct adaptively-secure IB-KEMs. We will use it to construct balanced PHFs with constant size keys, which then yields an IB-KEM with constant-size master public key and full adaptive security.

FULL RANK DIFFERENCE ENCODING FUNCTION FOR ECR HASH FUNCTIONS. We require a variant $\mathsf{FRD}^{\mathsf{ecr}}$ of Agrawal *et al.*'s full rank difference encoding function similar to the variant that we introduced for blockwise partitioning. However, in contrast to $\mathsf{FRD}^{\mathsf{blk}}$, $\mathsf{FRD}^{\mathsf{ecr}}$ just pads its input with zeros in the end instead of in the front and in the end.

**Definition 36.** For $\ell = \mathcal{O}(\log \lambda)$ and all $i \in [\ell]_0$ we define $\mathsf{FRD}_i^{\mathsf{ecr}} : \mathbb{Z}_q^{2^i} \to \mathbb{Z}_q^{n \times n}$ as the function that on input $\mathbf{a} = (a_1, \ldots, a_{2^i}) \in \mathbb{Z}_q^{2^i}$ outputs

$$\mathsf{FRD}_i^{\mathsf{ecr}}(\mathbf{a}) \coloneqq \mathsf{FRD}((a_1, \ldots, a_{2^i}, 0, \ldots, 0)).$$

Analogous to $\mathsf{FRD}^{\mathsf{blk}}$, the properties of $\mathsf{FRD}$ from Lemma 16 carry over to $\mathsf{FRD}^{\mathsf{ecr}}$. Essentially, it guarantees that we can map vectors injectively to matrices, such that the resulting matrices and their differences are invertible.

**Corollary 4.** *Let $\mathsf{FRD}_i^{\mathsf{ecr}} : \mathbb{Z}_q^{2^i} \to \mathbb{Z}_q^{n \times n}$ be as defined as above, then the following holds:*

1. *$\mathsf{FRD}_i^{\mathsf{ecr}}$ is injective.*
2. *There is an additive subgroup $\mathbb{G} \subset \mathbb{Z}_q^{n \times n}$ such that each $\mathbf{H} \in \mathbb{G} \setminus \{0\}$ is invertible and the range of $\mathsf{FRD}_i^{\mathsf{ecr}}$ is a subset of $\mathbb{G}$ for all $0 \leq i \leq \ell$.*

*Proof.* We first observe that padding $(a_1, \ldots, a_{2^i}) \in \mathbb{Z}_q^{2^i}$ with zeros construct a vector in $Z_q^n$ is an injective operation. Thus, since $\mathsf{FRD}$ is injective by Lemma 16 it follows that $\mathsf{FRD}_i^{\mathsf{ecr}}$ is also injective, which proves the first claim. Furthermore, since $\mathsf{FRD}_i^{\mathsf{ecr}}$ just pads its input with zeros and then outputs what $\mathsf{FRD}$ outputs, we have that the range of $\mathsf{FRD}_i^{\mathsf{ecr}}$ is necessarily a subset of the range of $\mathsf{FRD}$, which proves the second claim. □

ALPERIN SHERIFF'S METHOD. We proceed to describe Alperin Sheriff's approach, which our first balanced PHF from ECR hash functions is based on. For this, let $n(i)$ and $q(i)$ be as in Section 5.4.3. Then, let $\mathsf{id} = (\mathsf{id}_0, \ldots, \mathsf{id}_\ell)$, where $\mathsf{id}_i \in \mathbb{Z}_{q(i)}^{n(i)}$ for $0 \leq i \leq \ell$ for some integer $\ell \in \Theta(\log(\lambda))$, some $i^* \in [\ell]_0$ and some $\mathsf{K} \in \mathbb{Z}_{q(i^*)}^{2^{i^*}}$. We follow [Alp15] and define a function $f$ as

$$t_i \coloneqq \begin{cases} 1 & \text{if } i = i^* \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad f(\mathsf{id}, \mathsf{K}, i^*) \coloneqq -\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K}) + \sum_{i=0}^{\ell} t_i \mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i).$$

Note that we have

$$f(\mathsf{id}, \mathsf{K}, i^*) = \begin{cases} \mathbf{0} & \text{if } \mathsf{K} = \mathsf{id}_{i^*} \\ \mathbf{H}_{\mathsf{id}} \in \mathsf{GL}(\mathbb{Z}_q, n)_n & \text{otherwise} \end{cases} \tag{5.7}$$

because $f(\mathsf{K}, \mathsf{id}_{i^*}, i^*) = \mathbf{0}$ holds obviously and the "otherwise" condition follows immediately from the second property of Corollary 4.

The main idea of Alperin-Sheriff [Alp15] is to rewrite $f$ as follows. Consider the following degree-$\ell$ polynomials $p_i$ for all $0 \leq i \leq \ell$.

$$p_i(x) \coloneqq \begin{cases} 1 & \text{if } x = i, \\ 0 & \text{for } x \in \{0, \ldots, \ell\} \setminus \{i\} \end{cases}$$

Furthermore, let $c_{i,j}$ be coefficients such that

$$p_i(x) = \sum_{j=0}^{\ell} c_{i,j} x^j$$

holds. Observe that the definition of $p_i$ is independent of $i^*$ and $\mathsf{K}$ and therefore the coefficients $c_{i,j} \in \mathbb{Z}_q$ are *publicly* known. This allows us to rewrite $f(\mathsf{id}, \mathsf{K}, i^*)$ as

$$\begin{aligned} f(\mathsf{id}, \mathsf{K}, i^*) &= -\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K}) + \sum_{i=0}^{\ell} p_i(i^*) \mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i) \\ &= -\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K}) + \sum_{i=0}^{\ell} \mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i) \left( \sum_{j=0}^{\ell} c_{i,j} i^{*j} \right) \\ &= -\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K}) + \sum_{j=0}^{\ell} i^{*j} \left( \sum_{i=0}^{\ell} c_{ij} \mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i) \right). \end{aligned}$$

Intuition for the balanced PHF construction. The key feature of Alperin-Sheriff's method is that it makes it possible to define only *two* matrices $\mathbf{B}_{\mathsf{K}}$ and $\mathbf{B}$, such that by recursively applying arithmetic computations on these matrices one can compute a matrix $\mathbf{B}_{\mathsf{id}}$. Furthermore, and crucially, in a security proof it is additionally possible to *obliviously* embed $-\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K})$ and $i^*$ into $\mathbf{B}_{\mathsf{K}}$ and $\mathbf{B}$, respectively, such that by recursively applying these arithmetic computations one obtains

$$\mathbf{B}_{\mathsf{id}} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + f(\mathsf{id}, \mathsf{K}, i^*)\mathbf{G}$$

such that $f(\mathsf{id}, \mathsf{K}, i^*) = \mathbf{0}$ if $\mathsf{K} = \mathsf{id}_{i^*}$, and $f(\mathsf{id}, \mathsf{K}, i^*) \in \mathsf{GL}(\mathbb{Z}_q, n)$ otherwise. We will use this to construct lattice-based balanced PHFs.

**Construction 6.** For $i \in \mathbb{N}$ let $\mathcal{H}_{n(i)} = \{\{0,1\}^{\lambda} \to \mathbb{Z}_{q(i)}^{n(i)}\}$ be a family of $(c,d)$-ECR hash functions for constants $c > 1$ and $d > 0$, and let $\ell \coloneqq \lceil \log((c(\lambda+1)+1)/d) \rceil$. We start by describing the hash function $\mathcal{F}_{\mathsf{AS}} = (\mathsf{HGen}_{\mathsf{AS}}, \mathsf{HEval}_{\mathsf{AS}})$.

$\mathsf{HGen}_{\mathsf{AS}}(1^\lambda)$ samples $H_{\mathsf{ECR},i} \xleftarrow{\$} \mathcal{H}_{n(i)}$ for $0 \leq i \leq \ell$ and $\mathbf{B}_{\mathsf{K}}, \mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$. It then outputs $F_{\mathsf{AS}} := ((H_{\mathsf{ECR},i})_{0 \leq i \leq \ell}, \mathbf{B}_{\mathsf{K}}, \mathbf{B})$.

$\mathsf{HEval}_{\mathsf{AS}}(H_{\mathsf{AS}}, \mathsf{id})$ does the following:

1. Set $\mathbf{B}_1 := \mathbf{B}$ and $\mathbf{B}_j := \mathbf{B}_1 \mathbf{G}^{-1}(\mathbf{B}_{j-1})$ for all $2 \leq j \leq \ell$ and set $\mathsf{id}_i := H_i(\mathsf{id})$ for all $0 \leq i \leq \ell$.
2. Compute $\mathbf{H}_j := \sum_{i=0}^{\ell} c_{i,j} \mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)$ for all $0 \leq j \leq \ell$ and set $\mathbf{B}'_0 := \mathbf{H}_0 \mathbf{G}$.
3. Compute $\mathbf{B}'_j := \mathbf{B}_j \mathbf{G}^{-1}(\mathbf{H}_j \mathbf{G})$ for $1 \leq j \leq \ell$.
4. Compute and output $\mathbf{B}_{\mathsf{id}} := \mathbf{B}_{\mathsf{K}} + \sum_{j=0}^{\ell} \mathbf{B}'_j$.

Why this particular evaluation algorithm is useful will become clear with the description of the corresponding trapdoor generation and evaluation algorithms. Note that the description of the balanced PHF consists of two matrices $\mathbf{B}_{\mathsf{K}}, \mathbf{B}$ and $\ell + 1$ ECR hash functions. However, as shown in Lemma 21, when instantiated with the ECR family from Section 5.4.2, then these $\ell + 1$ hash functions can be represented by $\mathcal{O}(\lambda^2 \cdot \log(\lambda))$ many elements from $\{0, \ldots, q(\ell) - 1\}$. Hence, we have $|(H_{\mathsf{ECR},i})_{0 \leq i \leq \ell}| \in O(|\mathbb{Z}_q^{n \times m}|)$, where $|\mathbb{Z}_q^{n \times m}|$ is the size of the representation of a matrix in $\mathbb{Z}_q^{n \times m}$.

**Theorem 11.** *If $\mathcal{H}_{n(i)}$ is instantiated with the family of $(c, d)$-ECR hash functions from Construction 5 for $0 \leq i \leq \ell$, then $\mathcal{F}_{\mathsf{AS}} = (\mathsf{HGen}_{\mathsf{AS}}, \mathsf{HEval}_{\mathsf{AS}})$ is a $(\beta, \gamma, \delta)$-balanced PHF for $\beta := m + (\ell + 1)m^2\ell^\ell$, $\gamma = \mathsf{negl}(\lambda)$ and $\delta(t, \varepsilon) = \lambda^{-\mathcal{O}(\log\log(\lambda))}\varepsilon/2$, provided that $\mathsf{TrapGen}$ is only run on inputs $t, \varepsilon$ such that $t/\varepsilon < 2^\lambda$, where $t$ and $1/\varepsilon$ are polynomial in $\lambda$.*

*Proof.* We begin by describing the algorithms $\mathsf{TrapGen}$ and $\mathsf{TrapEval}$, which are based on the technique by Alperin-Sheriff [Alp15].

$\mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ works as follows:

1. It sets $i^* := \lceil \log((\log(t^c/\varepsilon) + c + 1)/d) \rceil$ and samples $\mathsf{K} \xleftarrow{\$} \mathbb{Z}_{q(i^*)}^{n(i^*)}$.
2. Sample $H_{\mathsf{ECR},i} \xleftarrow{\$} \mathcal{H}_{n(i)}$ for $0 \leq i \leq \ell$.
3. Sample $\mathbf{R}_{\mathsf{K}}, \mathbf{R} \xleftarrow{\$} \{-1, 1\}^{m \times m}$.
4. Set $\mathbf{B}_{\mathsf{K}} := \mathbf{A}\mathbf{R}_{\mathsf{K}} - \mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K})\mathbf{G}$ and $\mathbf{B} := \mathbf{A}\mathbf{R} + i^*\mathbf{G}$.
5. Return $F_{\mathsf{AS}} := ((H_{\mathsf{ECR},i})_{0 \leq i \leq \ell}, \mathbf{B}_{\mathsf{K}}, \mathbf{B})$ and $\mathsf{td} := (\mathbf{R}_{\mathsf{K}}, \mathbf{R}, \mathsf{K}, i^*)$.

$\mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{AS}}, \mathsf{id})$ does the following:

1. Set $\mathbf{B}_1 := \mathbf{B}$, $\mathbf{R}_1 := \mathbf{R}$, $\mathbf{R}_0 := \mathbf{0}$, and $\mathsf{id}_i := H_{\mathsf{ECR},i}(\mathsf{id})$, and compute $\mathbf{R}_j := \mathbf{R}\mathbf{G}^{-1}(\mathbf{B}_{j-1}) + i^*\mathbf{R}_{j-1}$ for $2 \leq j \leq \ell$.
2. Compute $\mathbf{H}_j := \sum_{i=0}^{\ell} c_{i,j} \mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)$ for $0 \leq j \leq \ell$, where the $c_{i,j}$ are the coefficients of the polynomials $p_i$ from Section 5.5.
3. Compute $\mathbf{R}'_j := \mathbf{R}_j \mathbf{G}^{-1}(\mathbf{H}_j \mathbf{G})$ for $1 \leq j \leq \ell$.

4. Return $\mathbf{R}_{\mathsf{id}} \coloneqq \mathbf{R}_{\mathsf{K}} + \sum_{j=1}^{\ell} \mathbf{R}'_j$ and $\mathbf{H}_{\mathsf{id}} \coloneqq f(\mathsf{id}, \mathsf{K}, i^*)$.

We first notice that $\mathsf{TrapGen}$ and $\mathsf{TrapEval}$ are syntactically correct and can be computed in probabilistic polynomial time. To show correctness, we need to prove that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$ and that $\mathsf{HEval}_{\mathsf{AS}}(F_{\mathsf{AS}}, \mathsf{id}) = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$ for $(F_{\mathsf{AS}}, \mathsf{td}) = \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}}) = \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{AS}}, \mathsf{id})$. We start by proving $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$.

Proving $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$. We bound the norm of $\mathbf{R}_{\mathsf{id}}$ by showing $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta = m + (\ell+1)m^2\ell^\ell$. We begin by showing that $\|\mathbf{R}_j\| \leq m\ell^\ell$ for all $1 \leq j \leq \ell$. As a preparation, we show that $\|\mathbf{R}_j\|_\infty \leq m \sum_{i=0}^{j-1} i^{*i}$ for all $1 \leq j \leq \ell$. Recall that we set $\mathbf{R}_1 \coloneqq \mathbf{R}$ and that thus $\|\mathbf{R}_1\|_\infty \leq 1 \leq m$ by definition of $\mathbf{R}$. Hence, we have by induction that for all $2 \leq j \leq \ell$ holds that

$$\begin{aligned}\|\mathbf{R}_j\|_\infty &= \left\|\mathbf{R}_1\mathbf{G}^{-1}(\mathbf{B}_{j-1}) + i^*\mathbf{R}_{j-1}\right\|_\infty \\ &\leq \left\|\mathbf{R}_1\mathbf{G}^{-1}(\mathbf{B}_{j-1})\right\|_\infty + i^* \left\|\mathbf{R}_{j-1}\right\|_\infty \\ &\leq m + i^* m \sum_{i=0}^{j-2} i^{*i} \leq m \sum_{i=0}^{j-1} i^{*i}.\end{aligned}$$

Applying the geometric series equality

$$\sum_{k=0}^{t} x^k = (x^{t+1} - 1)/(x - 1),$$

which holds for all $t \in \mathbb{N}$ and $x \neq 1$, we have, for $i^* \neq 1$, that

$$\|\mathbf{R}_j\|_\infty \leq m \sum_{i=0}^{j-1} i^{*i} = m\frac{i^{*j} - 1}{i^* - 1} \leq m i^{*j} \leq m\ell^\ell.$$

Moreover, if $i^* = 1$, then

$$m \sum_{i=0}^{j-2} i^{*i} = m \cdot j \leq m \cdot \ell^\ell$$

and the bound holds nonetheless. We proceed by bounding the norm of $\mathbf{R}'_j$.

$$\left\|\mathbf{R}'_j\right\|_\infty = \left\|\mathbf{R}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G})\right\|_\infty \leq \|\mathbf{R}_j\|_\infty m \leq m\ell^\ell m = m^2\ell^\ell$$

Note that the second inequality holds because $\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) \in \{0,1\}^{m \times m}$. To show that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq m + (\ell+1)m^3\ell^\ell$ we compute

$$\|\mathbf{R}_{\mathsf{id}}\|_\infty = \left\|\mathbf{R}_{\mathsf{K}} + \sum_{j=0}^{\ell} \mathbf{R}'_j\right\|_\infty \leq \|\mathbf{R}_{\mathsf{K}}\|_\infty + \sum_{j=0}^{\ell} \left\|\mathbf{R}'_j\right\|_\infty \leq m + (\ell+1)m^2\ell^\ell$$

Further, the trapdoor keys are statistically close because it holds that $(\mathbf{A}, \mathbf{B}_{\mathsf{K}})$ and $(\mathbf{A}, \mathbf{B}_1)$ have only a negligible statistical difference by the Leftover-Hash Lemma (see Lemma 15).

PROVING $\mathbf{B}_{\mathsf{id}} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$. To complete the proof of correctness, we show that Equation (5.2) holds. We first show that for all $1 \leq j \leq \ell$ we have that $\mathbf{B}_j = \mathbf{A}\mathbf{R}_j + i^{*j}\mathbf{G}$, where $\mathbf{B}_j$ and $\mathbf{R}_j$ are as specified by $\mathsf{HEval}_{\mathsf{AS}}$, $\mathsf{TrapGen}$, and $\mathsf{TrapEval}$. For $j = 1$, this holds trivially by definition of $\mathsf{HEval}_{\mathsf{AS}}$ and $\mathsf{TrapEval}$. By induction we obtain

$$\begin{aligned}\mathbf{B}_j &= \mathbf{B}_1\mathbf{G}^{-1}(\mathbf{B}_{j-1}) = (\mathbf{A}\mathbf{R}_1 + i^*\mathbf{G})\mathbf{G}^{-1}(\mathbf{B}_{j-1}) \\ &= \mathbf{A}\mathbf{R}_1\mathbf{G}^{-1}(\mathbf{B}_{j-1}) + i^*\mathbf{G}\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_{j-1} + i^{*j-1}\mathbf{G}) \\ &= \mathbf{A}(\mathbf{R}_1\mathbf{G}^{-1}(\mathbf{B}_{j-1}) + i^*\mathbf{R}_{j-1}) + i^*i^{*j-1}\mathbf{G} \\ &= \mathbf{A}\mathbf{R}_j + i^{*j}\mathbf{G}\end{aligned}$$

for all $1 \leq j \leq \ell$.

Next, we show that

$$\mathbf{B}'_j = \mathbf{A}\mathbf{R}'_j + i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\mathbf{G}$$

To this end, we calculate

$$\begin{aligned}\mathbf{B}'_j &= \mathbf{B}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) = (\mathbf{A}\mathbf{R}_j + i^{*j}\mathbf{G})\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) \\ &= \mathbf{A}\mathbf{R}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) + i^{*j}\mathbf{H}_j\mathbf{G} \\ &= \mathbf{A}\mathbf{R}'_j + i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\mathbf{G}.\end{aligned}$$

Finally, we obtain the correctness of $\mathcal{F}_{\mathsf{AS}}$ as follows:

$$\begin{aligned}\mathbf{B}_{\mathsf{id}} &= \mathbf{B}_{\mathsf{K}} + \sum_{j=0}^{\ell}\mathbf{B}'_j \\ &= \mathbf{A}\mathbf{R}_{\mathsf{K}} - \mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K})\mathbf{G} + \sum_{j=0}^{\ell}\left(\mathbf{A}\mathbf{R}'_j + i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\mathbf{G}\right) \\ &= \mathbf{A}\left(\mathbf{R}_{\mathsf{K}} + \sum_{j=0}^{\ell}\mathbf{R}'_j\right) + \left(-\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K}) + \sum_{j=0}^{\ell} i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\right)\mathbf{G} \\ &= \mathbf{A}\mathbf{R}_{\mathsf{id}} + f(\mathsf{id}, \mathsf{K}, i^*)\mathbf{G} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}, \qquad (5.8)\end{aligned}$$

where Equation (5.8) follows from the definition of $f(\mathsf{id}, \mathsf{K}, i^*)$ in Equation (5.7).

WELL-DISTRIBUTEDNESS. Let $\mathcal{A}$ be an algorithm that outputs $\mathcal{Q}^* := \{\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}, \mathsf{id}^*\}$ on input $F'_{\mathsf{AS}}$ with $(F'_{\mathsf{AS}}, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$. Furthermore, let

$$\begin{aligned}(\mathbf{R}_{\mathsf{id}^*}, \mathbf{H}_{\mathsf{id}^*}) &:= \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{AS}}, \mathsf{id}^*) \\ (\mathbf{R}_{\mathsf{id}^{(i)}}, \mathbf{H}_{\mathsf{id}^{(i)}}) &:= \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{AS}}, \mathsf{id}^{(i)}) \text{ for } 1 \leq i \leq Q\end{aligned}$$

and let $\mathsf{id}^{(Q+1)} := \mathsf{id}^*$ and the events $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}_{\mathsf{phf}}$ be defined as

$$\mathsf{coll}_{\mathsf{phf}} := \exists\, \mathsf{id} \neq \mathsf{id}' \in \mathcal{Q}^* : \mathbf{H}_{\mathsf{id}} = \mathbf{H}_{\mathsf{id}'}, \qquad \mathsf{badChal}_{\mathsf{phf}} := \mathbf{H}_{\mathsf{id}^*} \neq \mathbf{0}$$

as in Definition 31.

We begin by showing that $\Pr[\mathsf{coll}_{\mathsf{phf}}] \leq \varepsilon/2$. For this, observe that we have $\mathbf{H}_{\mathsf{id}} = \mathbf{H}'_{\mathsf{id}}$ *if and only if* $H_{\mathsf{ECR},i^*}(\mathsf{id}) = H_{\mathsf{ECR},i^*}(\mathsf{id}')$. This holds because $\mathsf{FRD}^{\mathsf{ecr}}_{i^*}$ is injective by Corollary 4 and because for all $\widehat{\mathsf{id}} \in \{0,1\}^\lambda$ and $\widehat{\mathsf{id}}_{i^*} := H_{\mathsf{ECR},i^*}(\widehat{\mathsf{id}})$ it holds that

$$\mathbf{H}_{\mathsf{id}} = f(\widehat{\mathsf{id}}, \mathsf{K}, i^*) = \mathsf{FRD}^{\mathsf{ecr}}_{i^*}(\widehat{\mathsf{id}}_{i^*}).$$

Hence, $\mathbf{H}_{\mathsf{id}} = \mathbf{H}_{\mathsf{id}'}$ implies a collision on $H_{\mathsf{ECR},i^*}$. Therefore we can describe an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the ECR of $\mathcal{H}_{i^*}$, provided that $\mathcal{A}$ produces a set $\mathcal{Q}^*$ such that $\mathsf{coll}_{\mathsf{phf}}$ occurs with probability larger than $\varepsilon/2$. More formally $\mathcal{B}$ receives as input

$$H_{\mathsf{ECR},i^*} \in \mathbb{Z}_{q(i^*)}^{n(i^*)\times m(i^*)}$$

and proceeds as follows.

1. Sample $\mathsf{K} \xleftarrow{\$} \mathbb{Z}_{q(i^*)}^{n(i^*)}$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $H_{\mathsf{ECR},i} \xleftarrow{\$} \mathcal{H}_{n_i}$ for all $0 \leq i \leq \ell$ with $i \neq i^*$.
2. Sample $\mathbf{R}_{\mathsf{K}}, \mathbf{R} \xleftarrow{\$} \{-1,1\}^{m\times m}$ and set $\mathbf{B}_{\mathsf{K}} := \mathbf{A}\mathbf{R}_{\mathsf{K}} + \mathsf{FRD}^{\mathsf{ecr}}_{i^*}(\mathsf{K})\mathbf{G}$ and $\mathbf{B} := \mathbf{A}\mathbf{R} + i^*\mathbf{G}$.
3. Set $F'_{\mathsf{AS}} := ((H_{\mathsf{ECR},i})_{0\leq i\leq \ell}, \mathbf{B}_{\mathsf{K}}, \mathbf{B})$ and compute $\mathcal{Q}^* = \{\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}, \mathsf{id}^*\} := \mathcal{A}(F'_{\mathsf{AS}})$.
4. If there are $\mathsf{id} \neq \mathsf{id}' \in \mathcal{Q}^*$ such that $H_{\mathsf{ECR},i^*}(\mathsf{id}) = H_{\mathsf{ECR},i^*}(\mathsf{id})$, then output $(\mathsf{id}, \mathsf{id}')$, otherwise abort.

Note that $\mathcal{B}$ essentially runs $\mathcal{A}$ plus some minor additional operations. Therefore, we upper-bound the running time of $\mathcal{B}$ by $t_{\mathcal{B}} := 2t$, where $t$ is the running time of $\mathcal{A}$. Observe that $F'_{\mathsf{AS}}$ is distributed exactly as if it was generated by $\mathsf{TrapGen}$. Hence, if $\mathcal{A}$ produces $\mathcal{Q}^*$ such that $\Pr[\mathsf{coll}_{\mathsf{phf}}] \geq \varepsilon/2$, then $\mathcal{B}$ outputs $(\mathsf{id}, \mathsf{id}')$ with $H_{\mathsf{ECR},i^*}(\mathsf{id}) = H_{\mathsf{ECR},i^*}(\mathsf{id})$ with probability at least $\varepsilon/2$. However, we have that

$$\frac{\varepsilon}{2} \geq \frac{t^c}{2^{dn-c}} = \frac{(2t)^c}{2^{dn}} = \frac{t_{\mathcal{B}}^c}{2^{dn}}$$

by Lemma 19 and thus $\mathcal{B}$ breaks the $(c,d)$-ECR of $\mathcal{H}_{i^*}$ if $\mathsf{coll}_{\mathsf{phf}}$ occurs with probability at least $\varepsilon/2$.

We next consider the event $\neg\mathsf{badChal}_{\mathsf{phf}}$. Observe that we have $\mathbf{H}_{\mathsf{id}^*} = \mathbf{0}$ if and only if $H_{\mathsf{ECR},i^*}(\mathsf{id}^*) = \mathsf{K}$ and that $\mathsf{TrapGen}$ samples $\mathsf{K} \xleftarrow{\$} \mathbb{Z}_{q(i^*)}^{n(i^*)}$. Thus, we conclude at this point that $\mathsf{badChal}_{\mathsf{phf}}$ and $\mathsf{coll}_{\mathsf{phf}}$ are independent because $\mathsf{K}$ is drawn uniformly at random from $\mathbb{Z}_{q(i^*)}^{n(i^*)}$. Furthermore, we thus have that

$$\Pr[\neg\mathsf{badChal}_{\mathsf{phf}}] = q(i^*)^{-n(i^*)}.$$

Now, we conclude that

$$(\varepsilon - \Pr[\mathsf{coll}_{\mathsf{phf}}]) \cdot \Pr[\neg\mathsf{badChal}_{\mathsf{phf}}] \leq \frac{\varepsilon}{2} q(i^*)^{-n(i^*)} = \frac{\varepsilon}{2}(\varepsilon/2)\lambda^{-\mathcal{O}(\log\log(\lambda))} = \delta(t,\varepsilon)$$

where $q(i^*)^{-n(i^*)} = \lambda^{-\mathcal{O}(\log\log(\lambda))}$ holds by Lemma 20. □

Discussion. The main advantage of this approach is that we obtain PHFs, which are balanced and whose key size is equivalent to a constant number of matrices in $\mathbb{Z}_q^{n\times m}$. The major downside of this approach is that the powers $i^{*2}, \ldots, i^{*\ell}$ have to be computed homomorphically, hidden from the adversary by $\mathsf{TrapEval}$. This has the consequence that we can only prove that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq m + (\ell+1)m^3\ell^\ell$. Unfortunately, the term $\ell^\ell$ is super-polynomial for our choice of $\ell$.

As Alperin-Sheriff [Alp15] discusses, this issue can be addressed to some degree by choosing $\omega(\log(\log(\lambda))) \leq \ell \leq \mathcal{O}(\log(\lambda)/\log(\log(\lambda)))$. However, even then $\|\mathbf{R}_{\mathsf{id}}\|_\infty$ is only bounded by a high-degree polynomial and the security holds only for $\lambda$ large enough.

We will therefore describe an alternative approach that allows us to replace the term $\ell^\ell$ in the upper bound of the norm by $\ell^{\lceil\sqrt{\ell}\rceil} = \lambda^{o(1)}$, at the cost of a (asymptotically) slightly larger description of the programmable hash function.

## 5.6 Balanced Programmable Hash Functions with Small-Norm Trapdoors

The naïve approach to reduce the norm of $\mathbf{R}_{\mathsf{id}}$ would be to pre-compute all powers of $i^*$ and embed each in a matrix in the description of the PHF. However, this would lead to $\ell = \mathcal{O}(\log(\lambda))$ many matrices in the description of the PHF and nullify the gains made by using Alperin-Sheriff's method in the first place.

A baby-step giant-step approach. We introduce a new approach, which follows a middle way by pre-computing only $2\lceil\sqrt{\ell}\rceil = \mathcal{O}(\sqrt{\log\lambda})$ different powers of $i^*$, allowing us to reduce $\ell^\ell$ to $\ell^{\lceil\sqrt{\ell}\rceil} = \lambda^{o(1)}$ for $\ell = \lceil\log((c(\lambda+1)+1)/d)\rceil$. For this purpose, we describe a list $\mathsf{S}_\ell$ of size $2\lceil\sqrt{\ell}\rceil$ and a function $\mathsf{pair}_\ell : \{1,\ldots,\ell\} \to \{0,\ldots,\ell\} \times \{0,\ldots,\ell\}$ such that for all $j \in \{1,\ldots,\ell\}$ and $\mathsf{pair}(j) = (j_1, j_2)$, it holds that $j_1, j_2 \in \mathsf{S}_\ell$ and $j = j_1 + j_2$. In particular, we then have $i^{*j} = i^{*j_1} \cdot i^{*j_2}$. Furthermore, we have that $j_2 \leq \lceil\sqrt{\ell}\rceil$, which allows us to reduce the norm of $\|\mathbf{R}_{\mathsf{id}}\|_\infty$ as outlined above.

Informally, we achieve this by a "baby-step giant-step" approach, where the list $\mathsf{S}_\ell$ contains all integers from zero up to $\lceil\sqrt{\ell}\rceil$ ("baby steps") and all integer multiples of $\lceil\sqrt{\ell}\rceil$ between zero and $\ell$ ("giant steps"). Then every number in $j \in [\ell]$ can be represented as the sum of the largest multiple of $\lceil\sqrt{\ell}\rceil$ that is smaller or equal than $j$ and a number between zero and $\lceil\sqrt{\ell}\rceil$. It will be convenient to formalize this in a lemma.

**Lemma 22.** *For $\ell \in \mathbb{N}$, let*

$$\mathsf{S}_\ell \coloneqq (1, \ldots, \lceil\sqrt{\ell}\rceil - 1) \quad \| \quad \left(m \cdot \lceil\sqrt{\ell}\rceil\right)_{0 \le m \le \lceil\sqrt{\ell}\rceil}$$

*where $||$ is concatenation. Then there is an polynomial-time computable function $\mathsf{pair}_\ell : \{1, \ldots, \ell\} \to \mathsf{S}_\ell \times \mathsf{S}_\ell$ such that for all $j \in \{1, \ldots, \ell\}$ and $(j_1, j_2) \coloneqq \mathsf{pair}_\ell(j)$ it holds that $j = j_1 + j_2$ and $j_2 \le \lceil\sqrt{\ell}\rceil$. In particular, we have that $|\mathsf{S}_\ell| = 2\lceil\sqrt{\ell}\rceil = \mathcal{O}(\sqrt{\ell})$.*

*Proof.* We have that $|\mathsf{S}_\ell| = \lceil\sqrt{\ell}\rceil - 1 + \lceil\sqrt{\ell}\rceil + 1 = 2\lceil\sqrt{\ell}\rceil = \mathcal{O}(\sqrt{\ell})$. We define the function $\mathsf{pair}$ as follows. For an input $j \in [\ell]$, it sets $m \coloneqq \left\lfloor j/\lceil\sqrt{\ell}\rceil \right\rfloor, j_1 \coloneqq m\lceil\sqrt{\ell}\rceil$ and $j_2 \coloneqq j - j_1$. Note that $j_1$ is the largest multiple of $\lceil\sqrt{\ell}\rceil$ smaller than $j$ and hence is contained in $\mathsf{S}_\ell$. It therefore holds that $0 \le j_2 \le \lceil\sqrt{\ell}\rceil$ and hence $j_2 \in \{1, \ldots, \lceil\sqrt{\ell}\rceil - 1\} \cup \{0 \cdot \lceil\sqrt{\ell}\rceil, 1 \cdot \lceil\sqrt{\ell}\rceil\}$. By the definition of $\mathsf{S}_\ell$, $j_2$ is therefore contained in $\mathsf{S}_\ell$. Also, $\mathsf{pair}_\ell$ can be efficiently evaluated. Finally, since $j_2 = j - j_1$, we also have that $j = j_1 + j_2$, which completes the proof. □

This approach allows us to construct an balanced PHF with still a very compact description but with much small bound on the norm of $\mathbf{R}_{\mathsf{id}}$.

**Construction 7.** For $i \in \mathbb{N}$ let $\mathcal{H}_{n(i)} = \{\{0,1\}^\lambda \to \mathbb{Z}_{q(i)}^{n(i)}\}$ be a family of $(c,d)$-ECR hash functions for some constants $c > 1$ and $d > 0$, and let $\ell \coloneqq \lceil\log((c(\lambda+1)+1)/d)\rceil$. We define a hash function $\mathcal{F}_{\mathsf{SN}} = (\mathsf{HGen}_{\mathsf{SN}}, \mathsf{HEval}_{\mathsf{SN}})$ as follows.

$\mathsf{HGen}_{\mathsf{SN}}(1^\lambda)$ runs $H_{\mathsf{ECR},i} \xleftarrow{\$} \mathcal{H}_{n(i)}$ for $0 \le i \le \ell$ and $\mathbf{B}_{\mathsf{K}}, \mathbf{B}_j \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $j \in S_\ell$ and outputs $F_{\mathsf{SN}} \coloneqq ((H_{\mathsf{ECR},i})_{0 \le i \le \ell}, \mathbf{B}_{\mathsf{K}}, (\mathbf{B}_j)_{j \in \mathsf{S}_\ell})$.

$\mathsf{HEval}_{\mathsf{SN}}(F_{\mathsf{SN}}, \mathsf{id})$ computes $(j_1, j_2) \coloneqq \mathsf{pair}_\ell(j)$ for all $j \notin \mathsf{S}_\ell$ and then:

1. It computes $\mathbf{B}_j \coloneqq \mathbf{B}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1})$ for all $2 \le j \le \ell$ with $j \notin \mathsf{S}_\ell$ and sets $\mathsf{id}_i \coloneqq H_{\mathsf{ECR},i}(\mathsf{id})$ for all $0 \le i \le \ell$.
2. $\mathbf{H}_j \coloneqq \sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)$ for all $0 \le j \le \ell$ and $\mathbf{B}'_0 \coloneqq \mathbf{H}_0\mathbf{G}$.
3. $\mathbf{B}'_j \coloneqq \mathbf{B}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G})$ for $1 \le j \le \ell$.
4. Finally, compute and output $\mathbf{B}_{\mathsf{id}} \coloneqq \mathbf{B}_{\mathsf{K}} + \sum_{j=0}^{\ell} \mathbf{B}'_j$.

The following theorem shows that this is indeed a balanced PHF.

**Theorem 12.** *If $\mathcal{H}_{n(i)}$ is instantiated by the family of $(c,d)$-ECR hash functions from Construction 5 for $0 \le i \le \ell$, then $\mathcal{F}_{\mathsf{SN}} = (\mathsf{HGen}_{\mathsf{SN}}, \mathsf{HEval}_{\mathsf{SN}})$ is a $(\beta, \gamma, \delta)$-balanced PHF for $\beta \coloneqq 1 + (\ell+1)m^2(1+\lambda^{o(1)}), \gamma = \mathsf{negl}(\lambda)$ and $\delta(t,\varepsilon) = \delta(t,\varepsilon) = \lambda^{-\mathcal{O}(\log\log(\lambda))}\varepsilon/2$, provided that $\mathsf{TrapGen}$ is only run on inputs $t, \varepsilon$ such that $t/\varepsilon < 2^\lambda$, where $t$ and $1/\varepsilon$ are polynomial in $\lambda$.*

*Proof.* We begin by describing the algorithms $\mathsf{TrapGen}$ and $\mathsf{TrapEval}$.

$\mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ sets $i^* \coloneqq \lceil\log((\log(t^c/\varepsilon) + c + 1)/d)\rceil$ and then:

1. Sample $\mathsf{K} \xleftarrow{\$} \mathbb{Z}_{q_{i^*}}^{n_{i^*}}$ and $H_{\mathsf{ECR},i} \xleftarrow{\$} \mathcal{H}_{n_i}$ for $0 \leq i \leq \ell$ and $\mathbf{R}_{\mathsf{K}}, \mathbf{R}_j \xleftarrow{\$} \{-1,1\}^{m \times m}$ for $j \in \mathsf{S}_\ell$.
2. Compute $\mathbf{B}_{\mathsf{K}} \coloneqq \mathbf{A}\mathbf{R}_{\mathsf{K}} - \mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K})\mathbf{G}$ and $\mathbf{B}_j \coloneqq \mathbf{R}_j + i^{*j}\mathbf{G}$ for all $j \in \mathsf{S}_\ell$.
3. Compute and output $F_{\mathsf{SN}} \coloneqq ((H_{\mathsf{ECR},i})_{0 \leq i \leq \ell}, \mathbf{B}_{\mathsf{K}}, (\mathbf{B}_j)_{j \in \mathsf{S}_\ell})$ and the trapdoor $\mathsf{td} \coloneqq (\mathbf{R}_{\mathsf{K}}, (\mathbf{R})_{j \in \mathsf{S}_\ell}, \mathsf{K}, i^*)$.

$\mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{SN}}, \mathsf{id})$ starts by computing $(j_1, j_2) \coloneqq \mathsf{pair}_\ell(j)$ and the matrix $\mathbf{R}_j \coloneqq \mathbf{R}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1}) + i^{*j_2}\mathbf{R}_{j_1}$ for all $j \notin \mathsf{S}_\ell$. It then does the following.

1. Compute $\mathbf{H}_j \coloneqq \sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_{n_i}^{\mathsf{ecr}}(H_{\mathsf{ECR},i}(\mathsf{id}))$ for $0 \leq j \leq \ell$.
2. Set $\mathbf{R}'_j \coloneqq \mathbf{R}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G})$ for $1 \leq j \leq \ell$ and $\mathbf{R}_{\mathsf{id}} \coloneqq \mathbf{R}_{\mathsf{K}} + \sum_{j=1}^{\ell}\mathbf{R}'_j$.
3. Finally, set $\mathbf{H}_{\mathsf{id}} \coloneqq f(\mathsf{id}, \mathsf{K}, i^*)$ and output $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}})$.

We first notice that $\mathsf{TrapGen}$ and $\mathsf{TrapEval}$ are syntactically correct and can be efficiently computed. For correctness we need to prove that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$ and that $\mathsf{HEval}_{\mathsf{SN}}(F_{\mathsf{SN}}, \mathsf{id}) = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$ for $(F_{\mathsf{SN}}, \mathsf{td}) = \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}}) = \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{SN}}, \mathsf{id})$. We start by proving $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$.

Proving $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq \beta$. We bound the norm of $\mathbf{R}_{\mathsf{id}}$ by showing that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq m + (\ell+1)(m^2 + m^2\ell^{\lceil\sqrt{\ell}\rceil})$. We begin by showing that $\|\mathbf{R}_j\| \leq m^2 + m\ell^{\lceil\sqrt{\ell}\rceil}$ for all $1 \leq j \leq \ell$. First, notice that $\|\mathbf{R}_j\|_\infty \leq 1$ for all $j \in \mathsf{S}_\ell$ because $\mathbf{R}_j \in \{-1,1\}^{m \times m}$ by definition of $\mathsf{TrapGen}$. For all $j \notin \mathsf{S}_\ell$, we have the following.

$$\begin{aligned}\|\mathbf{R}_j\|_\infty &= \left\|\mathbf{R}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1}) + i^{*j_2}\mathbf{R}_{j_1}\right\|_\infty \\ &\leq \left\|\mathbf{R}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1})\right\|_\infty + i^{*j_2}\|\mathbf{R}_{j_1}\|_\infty \\ &\leq m(1 + i^{*j_2})\end{aligned}$$

Note that the last inequality holds because $\mathbf{R}_{j_2} \in \{0,1\}^{m \times m}$ and $\mathbf{G}^{-1}(\mathbf{B}_{j_1}) \in \{0,1\}^m \times m$. Using that $j_2 \leq \lceil\sqrt{\ell}\rceil$ by Lemma 22 and $i^* \leq \ell$, we have that $\|\mathbf{R}_j\|_\infty \leq m^2 + m\ell^{\lceil\sqrt{\ell}\rceil}$. We proceed by bounding the norm of $\mathbf{R}'_j$.

$$\begin{aligned}\left\|\mathbf{R}'_j\right\|_\infty &= \left\|\mathbf{R}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G})\right\|_\infty \leq \|\mathbf{R}_j\|_\infty m \leq m(1 + i^{*j_2})m \\ &= m^2(1 + \ell^{\lceil\sqrt{\ell}\rceil})\end{aligned}$$

We now show that $\|\mathbf{R}_{\mathsf{id}}\|_\infty \leq 1 + (\ell+1)m^2(1 + \ell^{\lceil\sqrt{\ell}\rceil})$.

$$\begin{aligned}\|\mathbf{R}_{\mathsf{id}}\|_\infty &= \left\|\mathbf{R}_{\mathsf{K}} + \sum_{j=0}^{\ell}\mathbf{R}'_j\right\|_\infty \leq \|\mathbf{R}_{\mathsf{K}}\|_\infty + \sum_{j=0}^{\ell}\left\|\mathbf{R}'_j\right\|_\infty \\ &\leq 1 + (\ell+1)m^2(1 + \ell^{\lceil\sqrt{\ell}\rceil})\end{aligned}$$

To obtain the bound on $\|\mathbf{R}_{\mathsf{id}}\|_\infty$ claimed in the theorem, we need to show that $\ell^{\lceil\sqrt{\ell}\rceil} = \lambda^{o(1)}$ for $\ell = \lceil\lceil\log((c(\lambda+1)+1)/d)\rceil\rceil$. Ignoring constants and ceiling operations, this can be seen as follows:

$$\log(\lambda)^{\sqrt{\log(\lambda)}} = \lambda^{\log(\log(\lambda))\sqrt{\log(\lambda)}/\log(\lambda)} = \lambda^{\log(\log(\lambda))/\sqrt{\log(\lambda)}} = \lambda^{o(1)}.$$

However, the constants and ceiling operations make the proof rather tedious. We therefore state it as a lemma.

**Lemma 23.** *For $\ell = \lceil\lceil\log((c(\lambda+1)+1)/d)\rceil\rceil$ it holds that $\ell^{\lceil\sqrt{\ell}\rceil} = \lambda^{o(1)}$.*

To not interrupt the proof of Theorem 12, we postpone the proof of Lemma 23 until after we finished the proof of Theorem 12.

Applying Lemma 23 then yields that $1+(\ell+1)m^2(1+\ell^{\lceil\sqrt{\ell}\rceil}) \leq 1+(\ell+1)m^2(1+\lambda^{o(1)})$ as claimed.

Proving $\mathbf{B}_{\mathsf{id}} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$. To complete the proof of correctness, we show that Equation (5.2) holds. We first show that for all $0 \leq j \leq \ell$ we have that $\mathbf{B}_j = \mathbf{A}\mathbf{R}_j + i^{*j}\mathbf{G}$, where $\mathbf{B}_j$ and $\mathbf{R}_j$ are as specified by the algorithms $\mathsf{TrapGen}$ and $\mathsf{HEval}_{\mathsf{SN}}$. For all $j \in \mathsf{S}_\ell$, this holds by the definition of $\mathbf{B}_j$ in the $\mathsf{TrapGen}$ algorithm. For all $j \notin \mathsf{S}_\ell$, the matrix $\mathbf{B}_j$ is computed as follows by $\mathsf{HEval}_{\mathsf{SN}}$:

$$\begin{aligned}
\mathbf{B}_j &= \mathbf{B}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1}) = (\mathbf{A}\mathbf{R}_{j_2} + i^{*j_2}\mathbf{G})\mathbf{G}^{-1}(\mathbf{B}_{j_1}) \\
&= \mathbf{A}\mathbf{R}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1}) + i^{*j_2}\mathbf{G}\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_{j_1} + i^{*j_1}\mathbf{G}) \\
&= \mathbf{A}(\mathbf{R}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1}) + i^{*j_2}\mathbf{R}_{j_1}) + i^{*j_1+j_2}\mathbf{G} \\
&= \mathbf{A}(\mathbf{R}_{j_2}\mathbf{G}^{-1}(\mathbf{B}_{j_1}) + i^{*j_2}\mathbf{R}_{j_1}) + i^{*j}\mathbf{G}.
\end{aligned}$$

We next show that $\mathbf{B}'_j = \mathbf{A}\mathbf{R}'_j + i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\mathbf{G}$ holds for all $1 \leq j \leq \ell$

$$\begin{aligned}
\mathbf{B}'_j &= \mathbf{B}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) = (\mathbf{A}\mathbf{R}_j + i^{*j}\mathbf{G})\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) \\
&= \mathbf{A}\mathbf{R}_j\mathbf{G}^{-1}(\mathbf{H}_j\mathbf{G}) + i^{*j}\mathbf{H}_j\mathbf{G} \\
&= \mathbf{A}\mathbf{R}'_j + i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\mathbf{G}
\end{aligned}$$

Finally, we prove the correctness of $\mathcal{F}_{\mathsf{SN}}$, that is that $\mathbf{B}_{\mathsf{id}} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + f(\mathsf{id},\mathsf{K},i^*)\mathbf{G}$, where $\mathbf{B}_{\mathsf{id}}$ and $\mathbf{R}_{\mathsf{id}}$ are as defined by $\mathsf{TrapGen}$, $\mathsf{TrapEval}$ and $\mathsf{HEval}_{\mathsf{SN}}$.

$$\begin{aligned}
\mathbf{B}_{\mathsf{id}} &= \mathbf{B}_{\mathsf{K}} + \sum_{j=0}^{\ell}\mathbf{B}'_j \\
&= \mathbf{A}\mathbf{R}_{\mathsf{K}} - \mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K})\mathbf{G} + \sum_{j=0}^{\ell}\left(\mathbf{A}\mathbf{R}'_j + i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\mathbf{G}\right) \\
&= \mathbf{A}\left(\mathbf{R}_{\mathsf{K}} + \sum_{j=0}^{\ell}\mathbf{R}'_j\right) + \left(-\mathsf{FRD}_{i^*}^{\mathsf{ecr}}(\mathsf{K}) + \sum_{j=0}^{\ell} i^{*j}\left(\sum_{i=0}^{\ell} c_{i,j}\mathsf{FRD}_i^{\mathsf{ecr}}(\mathsf{id}_i)\right)\right)\mathbf{G} \\
&= \mathbf{A}\mathbf{R}_{\mathsf{id}} + f(\mathsf{id},\mathsf{K},i^*)\mathbf{G} = \mathbf{A}\mathbf{R}_{\mathsf{id}} + f(\mathsf{id},\mathsf{K},i^*)\mathbf{G} \qquad (5.9)
\end{aligned}$$

where Equation (5.9) follows from the definition of $f(\mathsf{id}, \mathsf{K}, i^*)$ in Equation (5.7).

Well-distributedness. The proof that $\mathcal{F}_{\mathsf{SN}}$ has well-distributed hidden matrices is almost identical to the respective proof for $\mathcal{F}_{\mathsf{AS}}$. Let $\mathcal{A}$ be an algorithm that outputs $\mathcal{Q}^* := \{\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}, \mathsf{id}^*\}$ on input $F'_{\mathsf{SN}}$ with $(F'_{\mathsf{SN}}, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ and $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$. Furthermore, let

$$(\mathbf{R}_{\mathsf{id}^*}, \mathbf{H}_{\mathsf{id}^*}) := \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{SN}}, \mathsf{id}^*)$$
$$(\mathbf{R}_{\mathsf{id}^{(i)}}, \mathbf{H}_{\mathsf{id}^{(i)}}) := \mathsf{TrapEval}(\mathsf{td}, F_{\mathsf{SN}}, \mathsf{id}^{(i)}) \text{ for } 1 \leq i \leq Q$$

and let $\mathsf{id}^{(Q+1)} := \mathsf{id}^*$ and let the events $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}_{\mathsf{phf}}$ be defined as

$$\mathsf{coll}_{\mathsf{phf}} := \exists\, \mathsf{id} \neq \mathsf{id}' \in \mathcal{Q}^* : \mathbf{H}_{\mathsf{id}} = \mathbf{H}_{\mathsf{id}'}, \qquad \mathsf{badChal}_{\mathsf{phf}} := \mathbf{H}_{\mathsf{id}^*} \neq \mathbf{0}$$

as in Definition 31.

We begin by showing that $\Pr[\mathsf{coll}_{\mathsf{phf}}] \leq \varepsilon/2$. For this, observe that we have $\mathbf{H}_{\mathsf{id}} = \mathbf{H}'_{\mathsf{id}}$ *if and only if* $H_{\mathsf{ECR},i^*}(\mathsf{id}) = H_{\mathsf{ECR},i^*}(\mathsf{id}')$. This holds because $\mathsf{FRD}^{\mathsf{ecr}}_{i^*}$ is injective by Corollary 4 and because for all $\widehat{\mathsf{id}} \in \{0,1\}^\lambda$ and $\widehat{\mathsf{id}}_{i^*} := H_{\mathsf{ECR},i^*}(\widehat{\mathsf{id}})$ it holds that

$$\mathbf{H}_{\mathsf{id}} = f(\widehat{\mathsf{id}}, \mathsf{K}, i^*) = \mathsf{FRD}^{\mathsf{ecr}}_{i^*}(\widehat{\mathsf{id}}_{i^*}).$$

Hence, $\mathbf{H}_{\mathsf{id}} = \mathbf{H}_{\mathsf{id}'}$ implies a collision on $H_{\mathsf{ECR},i^*}$. Therefore we can describe an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the ECR of $\mathcal{H}_{i^*}$, provided that $\mathcal{A}$ produces a set $\mathcal{Q}^*$ such that $\mathsf{coll}_{\mathsf{phf}}$ occurs with probability larger than $\varepsilon/2$. More formally $\mathcal{B}$ receives as input

$$H_{\mathsf{ECR},i^*} \in \mathbb{Z}_{q(i^*)}^{n(i^*) \times m(i^*)}$$

and proceeds as follows.

1. Sample $\mathsf{K} \xleftarrow{\$} \mathbb{Z}_{q(i^*)}^{n(i^*)}$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $H_{\mathsf{ECR},i} \xleftarrow{\$} \mathcal{H}_{n_i}$ for all $0 \leq i \leq \ell$ with $i \neq i^*$.
2. Sample $\mathbf{R}_{\mathsf{K}}, \mathbf{R} \xleftarrow{\$} \{-1,1\}^{m \times m}$ and set $\mathbf{B}_{\mathsf{K}} := \mathbf{A}\mathbf{R}_{\mathsf{K}} + \mathsf{FRD}^{\mathsf{ecr}}_{i^*}(\mathsf{K})\mathbf{G}$ and $\mathbf{B} := \mathbf{A}\mathbf{R} + i^*\mathbf{G}$.
3. Set $F'_{\mathsf{SN}} := ((H_{\mathsf{ECR},i})_{0 \leq i \leq \ell}, \mathbf{B}_{\mathsf{K}}, \mathbf{B})$ and compute $\mathcal{Q}^* = \{\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}, \mathsf{id}^*\} := \mathcal{A}(F'_{\mathsf{SN}})$.
4. If there are $\mathsf{id} \neq \mathsf{id}' \in \mathcal{Q}^*$ such that $H_{\mathsf{ECR},i^*}(\mathsf{id}) = H_{\mathsf{ECR},i^*}(\mathsf{id})$, then output $(\mathsf{id}, \mathsf{id}')$, otherwise abort.

Again $\mathcal{B}$ only runs $\mathcal{A}$ and performs some minor additional computations, so that we bound the running time of $\mathcal{B}$ by $t_{\mathcal{B}} := 2t$, where $t$ is the running time of $\mathcal{A}$. Observe that $F'_{\mathsf{SN}}$ is distributed exactly as if it was generated by $\mathsf{TrapGen}$. Hence, if $\mathcal{A}$ produces $\mathcal{Q}^*$ such that $\Pr[\mathsf{coll}_{\mathsf{phf}}] \geq \varepsilon/2$, then $\mathcal{B}$ outputs $(\mathsf{id}, \mathsf{id}')$ with $H_{\mathsf{ECR},i^*}(\mathsf{id}) = H_{\mathsf{ECR},i^*}(\mathsf{id})$ with probability at least $\varepsilon/2$. However, we have that

$$\frac{\varepsilon}{2} \geq \frac{t^c}{2^{dn-c}} = \frac{(2t)^c}{2^{dn}} = \frac{t_{\mathcal{B}}^c}{2^{dn}}$$

by Lemma 19 and thus $\mathcal{B}$ breaks the $(c,d)$-ECR of $\mathcal{H}_{i^*}$ if $\mathsf{coll}_{\mathsf{phf}}$ occurs with probability at least $\varepsilon/2$.

We next consider the event $\neg\mathsf{badChal}_{\mathsf{phf}}$. Observe that we have $\mathbf{H}_{\mathsf{id}^*} = \mathbf{0}$ if and only if $H_{\mathsf{ECR},i^*}(\mathsf{id}^*) = \mathsf{K}$ and that $\mathsf{TrapGen}$ samples $\mathsf{K} \xleftarrow{\$} \mathbb{Z}_{q(i^*)}^{n(i^*)}$. Thus, we conclude at this point that $\mathsf{badChal}_{\mathsf{phf}}$ and $\mathsf{coll}_{\mathsf{phf}}$ are independent because $\mathsf{K}$ is drawn uniformly at random from $\mathbb{Z}_{q(i^*)}^{n(i^*)}$. Furthermore, we thus have that

$$\Pr\left[\neg\mathsf{badChal}_{\mathsf{phf}}\right] = q(i^*)^{-n(i^*)}.$$

Now, we conclude that

$$(\varepsilon - \Pr\left[\mathsf{coll}_{\mathsf{phf}}\right]) \cdot \Pr\left[\neg\mathsf{badChal}_{\mathsf{phf}}\right] \leq \frac{\varepsilon}{2} q(i^*)^{-n(i^*)} = \frac{\varepsilon}{2}(\varepsilon/2)\lambda^{-\mathcal{O}(\log\log(\lambda))} = \delta(t,\varepsilon)$$

where $q(i^*)^{-n(i^*)} = \lambda^{-\mathcal{O}(\log\log(\lambda))}$ holds because of Lemma 20. □

Now it is only left to provide the proof of Lemma 23 that we omitted during the proof of Theorem 12.

*Proof of Lemma 23.* First, observe that there is a constant $a > 0$ such that $\ell \leq a\log(\lambda)$ for $\lambda$ large enough. We show that $\log(\lambda)^{\lceil\sqrt{\ell}\rceil} = \lambda^{o(1)}$ in preparation for the actual proof.

$$\begin{aligned}
\log(\lambda)^{\lceil\sqrt{\ell}\rceil} &= \left(\lambda^{\log(\log(\lambda))/\log(\lambda)}\right)^{\lceil\sqrt{\ell}\rceil} = \lambda^{\log(\log(\lambda))\lceil\sqrt{\ell}\rceil/\log(\lambda)} \\
&\leq \lambda^{\log(\log(\lambda))(\sqrt{a\log(\lambda)}+1)/\log(\lambda)} \\
&= \lambda^{\log(\log(\lambda))\sqrt{a\log(\lambda)}/\log(\lambda)}\lambda^{\log(\log(\lambda))/\log(\lambda)} \\
&= \lambda^{\log(\log(\lambda))\sqrt{a}/\sqrt{\log(\lambda)}}\lambda^{\log(\log(\lambda))/\log(\lambda)} \\
&= \lambda^{o(1)}\lambda^{o(1)} = \lambda^{o(1)}
\end{aligned}$$

Using this, we can easily conclude the lemma as follows.

$$\begin{aligned}
\ell^{\lceil\sqrt{\ell}\rceil} &\leq (a\log(\lambda))^{\lceil\sqrt{\ell}\rceil} = a^{\lceil\sqrt{\ell}\rceil}\log(\lambda)^{\lceil\sqrt{\ell}\rceil} = a^{\lceil\sqrt{\ell}\rceil}\lambda^{o(1)} \\
&\leq \log(\lambda)^{\lceil\sqrt{\ell}\rceil}\lambda^{o(1)} = \lambda^{o(1)}\lambda^{o(1)} = \lambda^{o(1)}
\end{aligned}$$

Note that we use that $a \leq \log(\lambda)$ for $\lambda$ large enough. □

## 5.7 Efficient Lattice-Based Identity-Based Key-Encapsulation

We proceed to describe our IB-KEM based on balanced PHFs. Our construction is based on the identity-based encryption scheme from [Yam17a]. We do not base our construction on the less efficient PHF-based IBE from Zhang *et al.* [ZCZ16], which also additionally requires that the PHF has high min-entropy. Throughout this section, we first introduce some further preliminaries on lattices before we describe the construction and prove its correctness and security.

### 5.7.1 Preliminaries for Lattice-Based IB-KEMs

We provide some further preliminaries that we require only for the construction and security proof of the IB-KEM. Let $m > 0$ be an integer and let $D_{\mathbb{Z}^m,\sigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter $\sigma > 0$. Further, recall that we denote the largest absolute component of a matrix by $\|\cdot\|_\infty$ and the $\ell_2$-norm of a matrix by $\|\cdot\|_2$. We then have the following lemma due to Regev [Reg05].

**Lemma 24.** *We have* $\Pr\left[\|x\|_2 > \sigma\sqrt{m} : \mathbf{x} \xleftarrow{\$} D_{\mathbb{Z}^m,\sigma}\right] \leq 2^{-2m}$.

The learning with errors problem. The security of our IB-KEM is based on the *learning with errors* (LWE) problem, which was introduced by Regev [Reg05]. We formally introduce it below.

**Definition 37.** Let $n = n(\lambda), m = m(\lambda)$ be integers, $q = q(n) > 2$ be prime, and $\alpha \in (0,1)$ be a real number. The advantage in solving the learning with errors problem $\mathsf{dLWE}_{n,m,q,\alpha}$ of a PPT algorithm $\mathcal{A}$ is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dLWE}_{n,m,q,\alpha}}(\lambda) := \left|\Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{x}^T) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{w}^T + \mathbf{x}^T) = 1\right]\right|,$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{x} \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha q}, \mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^m$ and the probability is over the random choices of $\mathbf{A}, \mathbf{s}, \mathbf{x}$ and $\mathbf{w}$, and the internal randomness of $\mathcal{A}$. Then for $t \in \mathbb{N}$ and $\varepsilon \in (0,1]$, the $(t,\varepsilon)$-$\mathsf{dLWE}_{n,m,q,\alpha}$ assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dLWE}_{n,m,q,\alpha}}(\lambda) \leq \varepsilon$ for all probabilistic algorithms $\mathcal{A}$ running in time at most $t$.

It is known that solving $\mathsf{dLWE}_{n,m,q,\alpha}$ for $\alpha q \geq 2\sqrt{n}$ is (quantumly) at least as hard as solving worst-case lattice problems $\mathsf{GapSVP}_\gamma$ and $\mathsf{SIVP}_\gamma$ on $n$-dimensional lattices for some $\gamma = \tilde{\mathcal{O}}(n/\alpha)$ [Reg05, Pei09, BLP+13]. Moreover, Katsumata and Yamada [KY16] proved the following lemma that enables rerandomizing LWE instances.

**Lemma 25.** *Let $q, m, m'$ be positive integers and $r$ a positive real satisfying $r > \max\{\omega(\sqrt{\log(m)}), \omega(\sqrt{\log(m')})\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{x} \xleftarrow{\$} D_{\mathbb{Z}^m,r}$. Then for any $\mathbf{V} \in \mathbb{Z}^{m\times m'}$ and positive real $s > \|\mathbf{V}\|_2$, there exists a PPT algorithm $\mathsf{ReRand}(\mathbf{V}, \mathbf{b}+\mathbf{x}, r, s)$ that outputs $\mathbf{b}' = \mathbf{b}^T\mathbf{V} + \mathbf{x}'^T \in \mathbb{Z}_q^{m'\times 1}$, where $\mathbf{x}'$ is distributed statistically close to $D_{\mathbb{Z}^{m'},2rs}$.*

Lattice trapdoors. In addition to the gadget matrix, which we introduced in Lemma 13, we use the following lattice trapdoors in the construction of our IB-KEM. Following Brakerski and Vaikuntanathan [BV16], let $n, m, m', q \in \mathbb{N}$ and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and let $\mathbf{A}_\sigma^{-1}(\mathbf{V})$ with $\mathbf{V} \in \mathbb{Z}_q^{n\times m'}$ denote the random variable whose distribution is a Gaussian $D_{\mathbb{Z}^m,\sigma}^{m'}$ conditioned on $\mathbf{A}\cdot\mathbf{A}_\sigma^{-1}(\mathbf{V}) = \mathbf{V}$. A *$\sigma$-trapdoor* for $\mathbf{A}$ is an algorithm that samples from $\mathbf{A}_\sigma^{-1}(\mathbf{V})$ in time $\mathsf{poly}(n, m, m', \log(q))$ for any $\mathbf{V} \in \mathbb{Z}_q^{n,\times m'}$. Note that we again slightly overload notation and denote a $\sigma$-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\sigma^{-1}$.

**Lemma 26.** *The following claims were proven in [GPV08, ABB10a, ABB10b, CHKP10, BLP+13].*

1. *There exists an efficient algorithm* $\mathsf{GenTrap}(1^n, 1^m, q)$ *that outputs* $\left(\mathbf{A}, \mathbf{A}_{\sigma_0}^{-1}\right)$ *for some* $m = \mathcal{O}(n \log(q))$*, such that* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *is* $2^{-n}$*-close to a uniformly random matrix and* $\sigma_0 = \omega(\sqrt{n \log q \log m})$.
2. *Given* $\mathbf{A}_\sigma^{-1}$*, one can obtain* $\mathbf{A}_{\sigma'}^{-1}$ *for any* $\sigma' \geq \sigma$.
3. *Given* $\mathbf{A}_\sigma^{-1}$*, one can obtain* $[\mathbf{A} \mid \mathbf{B}]_\sigma^{-1}$ *for any* $\mathbf{B}$.
4. *For* $\mathbf{A}_\sigma^{-1}$ *and* $\mathbf{u} \in \mathbb{Z}_q^n$*, it holds* $\Pr\left[\|\mathbf{A}_\sigma^{-1}\|_2 > \sqrt{m} \cdot \sigma\right] = \mathsf{negl}(\lambda)$.

## 5.7.2 Construction of the IB-KEM

Given these preliminaries, we are ready to describe our IB-KEM that is based on the IB-KEM by Yamada [Yam17a].

**Construction 8.** Let $\mathcal{F} = (\mathsf{HGen}, \mathsf{HEval})$ be a matrix hash function with range $\mathcal{T} = \mathbb{Z}_q^{n \times m}$. We construct $\mathcal{IB}\text{-}\mathcal{KEM} = (\mathsf{SetupIBK}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ as follows.

$\mathsf{SetupIBK}(1^\lambda)$ chooses $n, m, q, \ell, \sigma, \alpha$ and $\alpha'$ as specified in Remark 12 such that $q$ is a prime. It then runs $(\mathbf{A}, \mathbf{A}_{\sigma_0}^{-1}) \xleftarrow{\$} \mathsf{GenTrap}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\sigma_0 = \omega(\sqrt{n \log(q) \log(m)})$ holds. Next, it samples $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ and $F \xleftarrow{\$} \mathsf{HGen}(1^\lambda)$. Finally, it outputs

$$\mathsf{mpk} \coloneqq (\mathbf{A}, \mathbf{C}, F, \mathbf{u}) \qquad \text{and} \qquad \mathsf{msk} \coloneqq \mathbf{A}_{\sigma_0}^{-1}.$$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$ computes $\mathbf{B}_{\mathsf{id}} \coloneqq \mathsf{HEval}(F, \mathsf{id})$ with $\mathbf{B}_{\mathsf{id}} \in \mathbb{Z}_q^{n \times m}$. It then uses $\mathbf{A}_{\sigma_0}^{-1}$ to compute $[\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}}]_\sigma^{-1}$ and samples

$$\mathbf{e} \xleftarrow{\$} [\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}}]_\sigma^{-1}.$$

It returns $\mathsf{sk}_{\mathsf{id}} \coloneqq \mathbf{e} \in \mathbb{Z}^{2m}$. Observe that $[\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}}]\mathbf{e} = \mathbf{u} \bmod q$.

$\mathsf{Encap}(\mathsf{id}, \mathsf{mpk})$ computes $\mathbf{B}_{\mathsf{id}} \coloneqq \mathsf{HEval}(F, \mathsf{id}) \in \mathbb{Z}_q^{n \times m}$. It samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $x_0 \xleftarrow{\$} D_{\mathbb{Z}, \alpha q}$, $\mathbf{x}_1, \mathbf{x}_2 \xleftarrow{\$} D_{\mathbb{Z}^m, \alpha' q}$, and $\kappa \xleftarrow{\$} \{0,1\}$ and computes

$$c_0 = \mathbf{s}^\mathsf{T}\mathbf{u} + x_0 + \kappa \cdot \lceil q/2 \rceil \in \mathbb{Z}_q, \quad \mathbf{c}_1^\mathsf{T} = \mathbf{s}^\mathsf{T}[\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}}] + [\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}] \in \mathbb{Z}_q^{2m}.$$

It outputs $(\mathsf{ct} = (c_0, \mathbf{c}_1), \kappa)$.

$\mathsf{Decap}(\mathsf{ct}, \mathsf{usk}_{\mathsf{id}})$ parses $\mathsf{ct}$ as $(c_0, \mathbf{c}_1)$ and computes $w = c_0 - \mathbf{c}_1^\mathsf{T} \cdot \mathbf{e} \in \mathbb{Z}_q$. It sets $\kappa \coloneqq 1$ if $|w - \lceil q/2 \rceil| < \lceil q/4 \rceil$ and $\kappa \coloneqq 0$ otherwise. It then returns $\kappa$.

*Remark* 12. For the IB-KEM we have the following requirements regarding the parameter choice similar to [Yam17b]:

- As we show below, we require that $q/5 \geq \alpha q\sqrt{m} + (\alpha'\sqrt{2m}) \cdot \sigma\sqrt{2m}$ holds with overwhelming probability in order to guarantee the correctness of the scheme;
- That $\mathsf{GenTrap}$ can operate, that is $m > 6n\lceil \log q \rceil$;
- That the leftover hash lemma (Lemma 15) can be applied, meaning $m \geq (n+1)\log(q) + \omega(\log(n))$;
- $\sigma$ has to be large enough such that the distribution of private keys in the actual scheme and in the reduction is the same, that is $\sigma > \sigma_0 = \omega(\sqrt{n\log(q)\log(m)})$ and $\sigma > m(\beta+1)\omega(\sqrt{\log(m)})$;
- That the $\mathsf{ReRand}$ algorithm can operate in the reduction, that is $\alpha'/2\alpha > \sqrt{2} \cdot m(\beta+1)$ and $\alpha q > \omega(\sqrt{\log(m)})$ by Lemma 25;
- That the worst to average case reduction works, that is $\alpha q > 2\sqrt{2n}$.

To satisfy the above requirements, we set the parameters as follows:

$$\begin{aligned} &n = \Theta(\lambda), && m = \Theta(n\log(q)), && q = n^{9/2} \cdot \beta^2\omega(\log^{7/2}(n)) \\ &\sigma = m \cdot (\beta+1) \cdot \omega(\sqrt{\log(m)}) && \alpha q = 3\sqrt{n}, && \alpha' q = 9\sqrt{n} \cdot m \cdot (\beta+1). \end{aligned}$$

Furthermore, to instantiate the IB-KEM with one of the balanced PHFs from ECR hash functions, we also require that

$$n \geq 2^{\ell} \qquad \text{and} \qquad q \geq q(\ell),$$

where $\ell = \lceil \log((c(\lambda+1)+1)/d) \rceil$ and $c$ and $d$ are constants such that there is a family of $(c,d)$-ECR hash functions like the one that we describe in Construction 5. If $\mathcal{IB}\text{-}\mathcal{KEM}$ is instantiated with $\mathcal{F}_{\mathsf{BLK}}$ from Construction 4 based on a weak near-collision resistant hash functions, then we instead only require that

$$n \geq 2\lambda + 3.$$

**Correctness.**

We first deduce the error term as in [Yam17b]. For a ciphertext $\mathsf{ct} = (c_0, \mathbf{c}_1)$, let $\mathsf{usk}_{\mathsf{id}} = \mathbf{e}$ and $w$ be as described in $\mathsf{Decap}(\mathsf{ct}, \mathsf{usk}_{\mathsf{id}})$. We then have that

$$w = c_0 - \mathbf{c}_1^{\mathsf{T}} \cdot \mathbf{e} = \kappa \cdot \lceil q/2 \rceil + \underbrace{x_0 - \left[\mathbf{x}_1^{\mathsf{T}} \mid \mathbf{x}_2^{\mathsf{T}}\right] \cdot \mathbf{e}}_{:=\mathsf{err}},$$

holds, where we refer to $\mathsf{err} \coloneqq x_0 - \left[\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}\right] \cdot \mathbf{e}$ as the error term. Assuming $\alpha' \geq \alpha$, as guaranteed by our parameter choice in Remark 12, we then have that

$$\begin{aligned} |\mathsf{err}| &\leq |x_0| + \left|\left[\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}\right] \cdot \mathbf{e}\right| && (5.10)\\ &\leq |x_0| + \left\|\left[\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}\right]\right\|_2 \cdot \|\mathbf{e}\|_2 && (5.11)\\ &\leq \alpha q\sqrt{m} + (\alpha'\sqrt{2m}) \cdot \sigma\sqrt{2m} && (5.12)\\ &= \mathcal{O}(\alpha'\sigma m q) \end{aligned}$$

holds with overwhelming probability. Equation (5.10) follows from the triangle inequality, Equation (5.11) follows from the Cauchy-Schwartz inequality, and Equation (5.12) follows from Lemma 24 and Item 4 of Lemma 26. Note that the bound does not hold with certainty because Lemma 24 and Item 4 of Lemma 26 only guarantee that their respective bounds hold with probability $1 - \mathsf{negl}(\lambda)$. However, we treat the bounds as if they hold with certainty for the remainder of this discussion for simplicity.

Now, let $q > \Omega(\alpha'\sigma m q)$ such that $|\mathsf{err}| < q/5$ holds. We then have that

$$|w - \lceil q/2 \rceil| = |\kappa \lceil q/2 \rceil + \mathsf{err} - \lceil q/2 \rceil| \leq |\kappa \lceil q/2 \rceil - \lceil q/2 \rceil| + q/5$$

holds and we can thus conclude that

$$|\kappa \lceil q/2 \rceil - \lceil q/2 \rceil| + q/5 = \begin{cases} ||\lceil q/2 \rceil - \lceil q/2 \rceil| + q/5 = q/5 < \lceil q/4 \rceil & \text{if } \kappa = 1 \text{ and} \\ |-\lceil q/2 \rceil| + q/5 \geq 7q/10 \geq \lceil q/4 \rceil & \text{otherwise.} \end{cases}$$

Thus, $\mathsf{Decap}$ outputs the correct $\kappa$ with overwhelming probability.

### Multi-bit Variant of our IB-KEM.

The IB-KEM we described above can be adapted to encapsulate $\lambda$ many bits with a reasonable overhead. This can be achieved as described by Yamada [Yam17a, Section 5.3] by applying the techniques of [PVW08, ABB10a, Yam16]. Specifically, in order to encapsulate keys of $\lambda$ many bits, the vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ in the master public key has to be replaced with a matrix $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times \lambda}$. The ciphertext component $c_0 \in \mathbb{Z}_q$ is then replaced by a vector $\mathbf{c}_0^\mathsf{T} \in \mathbb{Z}_q^\lambda$ that is computed as

$$\mathbf{c}_0^\mathsf{T} \coloneqq \mathbf{s}^\mathsf{T}\mathbf{U} + \mathbf{x}_0^\mathsf{T} + \kappa \lceil q/2 \rceil ,$$

where $\mathbf{x}_0 \xleftarrow{\$} D_{\mathbb{Z}^\lambda, \alpha q}$ and $\kappa \xleftarrow{\$} \{0,1\}^\lambda$. Furthermore, the user secret key $\mathsf{usk} = \mathbf{e} \in \mathbb{Z}^{2m}$ has to be replaced by the matrix $\mathbf{E} \in \mathbb{Z}^{m \times \lambda}$ for

$$\mathbf{E} \xleftarrow{\$} [\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}}]_\sigma^{-1}(\mathbf{U}).$$

The security of the scheme can then be proven along the lines of the proof of Theorem 13 under the $\mathsf{dLWE}_{n,m+\lambda,q,\alpha}$ assumption. The parameter selection from Remark 12 remains valid. Overall, the asymptotic sizes of $\mathsf{mpk}$ and $\mathsf{ct}$ stay the same because $\mathsf{mpk}$ already contains at least the matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ and because $\mathsf{ct}$ already contains $\mathbf{c}_1^\mathsf{T} \in \mathbb{Z}_q^{2m}$. Only the size of user secret keys increases by a factor of $\lambda$.

### IND-ID-CPA security of $\mathcal{IB}$-$\mathcal{KEM}$

**Theorem 13.** *If $\mathcal{IB}$-$\mathcal{KEM} \coloneqq (\mathsf{SetupIBK}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ from above is instantiated with a $(\beta, \gamma, \delta)$-balanced programmable hash function with $\gamma = \mathsf{negl}(\lambda)$, then $\mathcal{IB}$-$\mathcal{KEM}$ is $(t, \varepsilon)$-IND-ID-CPA secure under the $(t, \varepsilon')$-$\mathsf{dLWE}_{n,m+1,q,\alpha}$ assumption for $\varepsilon' \coloneqq \delta(t, \varepsilon) - \mathsf{negl}(\lambda)$.*

*Proof.* We prove Theorem 13 in a sequence of games [Sho04]. We denote with $\mathsf{G}_i$ the event that Game $i$ outputs 1 and with $\mathsf{E}_i \coloneqq |\Pr[1 \xleftarrow{\$} \mathsf{G}_i] - 1/2|$ the advantage of $\mathcal{A}$ in Game $i$. The proof essentially adopts the proof from [Yam17a] to balanced programmable hash functions.

*Game 0.* This is the original IND-ID-CPA security experiment. Therefore, we have

$$\mathsf{E}_0 = \mathsf{Adv}^{\mathsf{IND\text{-}ID\text{-}CPA}}_{\mathcal{A},\mathcal{IB}\text{-}\mathcal{KEM}}(\lambda).$$

*Game 1.* This game is identical to Game 0, except that the challenger computes $(F', \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ in addition to $F \xleftarrow{\$} \mathsf{HGen}(1^\lambda)$. Note that the challenger still sets $\mathsf{mpk} \coloneqq (\mathbf{A}, \mathbf{C}, F, \mathbf{u})$ as specified in $\mathsf{SetupIBK}$ and does not include $F'$ in $\mathsf{mpk}$. Therefore, this is a purely conceptual change and we thus have that

$$\mathsf{E}_1 = \mathsf{E}_0.$$

*Game 2.* This game is identical to Game 1, except that the challenger maintains a set $\mathcal{Q}^* \coloneqq \{\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}, \mathsf{id}^*\}$, where $\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(Q)}$ are all the identities for which the adversary made queries to the $\mathsf{KeyGen}$ oracle and $\mathsf{id}^*$ is the challenge identity chosen by $\mathcal{A}$. It then computes $(\mathbf{R}_{\mathsf{id}^*}, \mathbf{H}_{\mathsf{id}^*}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F', \mathsf{id}^*)$ and $(\mathbf{R}_{\mathsf{id}^{(i)}}, \mathbf{H}_{\mathsf{id}^{(i)}}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F', \mathsf{id}^{(i)})$ for all $1 \leq i \leq Q$. Further, it defines the events $\mathsf{coll}_{\mathsf{phf}}$ and $\mathsf{badChal}_{\mathsf{phf}}$ as

$$\mathsf{coll}_{\mathsf{phf}} \coloneqq \exists\, \mathsf{id} \neq \mathsf{id}' \in \mathcal{Q}^* : H_{\mathsf{id}} = H_{\mathsf{id}'}, \qquad \mathsf{badChal}_{\mathsf{phf}} \coloneqq H_{\mathsf{id}^*} \neq \mathbf{0}$$

as in Definition 31. This is a purely conceptual change and it thus holds that

$$\mathsf{E}_2 = \mathsf{E}_1.$$

*Game 3.* This game is identical to Game 2, except that the challenger aborts and outputs a random bit if event $\mathsf{coll}_{\mathsf{phf}}$ occurs. It thus holds that

$$\begin{aligned}
\Pr[\mathsf{coll}_{\mathsf{phf}}] &\geq |\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3]| && (5.13)\\
&= \left|\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3] + \frac{1}{2} - \frac{1}{2}\right| \\
&= \left|\left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_3] - \frac{1}{2}\right)\right| \\
&= \left|\left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_3] - \frac{1}{2}\right)\right| + \left|\Pr[\mathsf{G}_3] - \frac{1}{2}\right| - \left|\Pr[\mathsf{G}_3] - \frac{1}{2}\right| \\
&\geq \left|\left(\Pr[\mathsf{G}_2] - \frac{1}{2}\right) - \left(\Pr[\mathsf{G}_3] - \frac{1}{2}\right) + \left(\Pr[\mathsf{G}_3] - \frac{1}{2}\right)\right| - \left|\Pr[\mathsf{G}_3] - \frac{1}{2}\right| && (5.14)\\
&= \left|\Pr[\mathsf{G}_2] - \frac{1}{2}\right| - \left|\Pr[\mathsf{G}_3] - \frac{1}{2}\right| = \mathsf{E}_2 - \mathsf{E}_3,
\end{aligned}$$

where Equation (5.13) follows from Shoup's Difference Lemma (see Lemma 1) and Equation (5.14) follows from the triangle inequality. We thus obtain that

$$\mathsf{E}_3 \geq \mathsf{E}_2 - \Pr[\mathsf{coll}_{\mathsf{phf}}]$$

holds by rearranging terms.

*Game 4.* This game is identical to Game 3, except that the challenger aborts and outputs a random bit if the event $\mathsf{badChal}_{\mathsf{phf}}$ occurs. We have that

$$\mathsf{E}_4 = \mathsf{E}_3 \cdot \Pr[\mathsf{badChal}_{\mathsf{phf}}]$$

holds because $\mathsf{badChal}_{\mathsf{phf}}$ is independent of the adversaries success in Game 3 since no information about the PHF with the trapdoor is given to the adversary in both the previous game and also this game and because $\mathsf{badChal}_{\mathsf{phf}}$ is also independent of $\mathsf{coll}_{\mathsf{phf}}$ by the well-distributedness of the balanced PHF.

*Game 5.* This game is identical to Game 4, except that we introduce the event $\mathsf{badEval}^{\mathsf{phf}}$ that occurs if the adversary makes a secret key query $\mathsf{id}$ such that $\mathbf{H}_{\mathsf{id}} = \mathbf{0}$. If $\mathsf{badEval}^{\mathsf{phf}}$ occurs, then the challenger aborts. Observe that this is a purely conceptual change because if $\mathsf{badEval}^{\mathsf{phf}}$ occurs than either $\mathsf{coll}_{\mathsf{phf}}$ or $\mathsf{badChal}_{\mathsf{phf}}$ *must occur*. This is because if event $\mathsf{badEval}^{\mathsf{phf}}$ occurs and $\mathsf{coll}_{\mathsf{phf}}$ does not occur then $\mathbf{H}_{\mathsf{id}^*} \neq \mathbf{0}$ *can not* hold since this would imply a collision, thus causing the event $\mathsf{badChal}_{\mathsf{phf}}$. Analogously, if event $\mathsf{badEval}^{\mathsf{phf}}$ occurs and $\mathsf{badChal}_{\mathsf{phf}}$ does not occur then event $\mathsf{coll}_{\mathsf{phf}}$ must happen because we then have that $\mathbf{H}_{\mathsf{id}} = \mathbf{0} = \mathbf{H}_{\mathsf{id}^*}$. it thus holds that

$$\mathsf{E}_5 = \mathsf{E}_4.$$

*Game 6.* This game is identical to Game 5, except that the challenger no longer computes $F \xleftarrow{\$} \mathsf{HGen}(1^\lambda)$ but sets $F \coloneqq F'$, where $(F', \mathsf{td}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda, t, \varepsilon, \mathbf{A})$ as specified in Game 1. Since $\mathcal{F}$ has statically close keys, we have that

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

holds, where the last equality holds because we required $\gamma = \mathsf{negl}(\lambda)$.

*Game 7.* This game is identical to Game 6, except that whenever the adversary queries the $\mathsf{KeyGen}$-oracle for $\mathsf{id}$, then the challenger computes $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}}) = \mathsf{TrapEval}(\mathsf{td}, F, \mathsf{id})$ and then sets $\mathbf{B}_{\mathsf{id}} \coloneqq \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$. This is a purely conceptual change since we have that $\mathsf{HEval}(F, \mathsf{id}) = \mathbf{A}\mathbf{R}_{\mathsf{id}} + \mathbf{H}_{\mathsf{id}}\mathbf{G}$ by the correctness of the balanced PHF $\mathcal{F}$. It thus holds that

$$\mathsf{E}_7 = \mathsf{E}_6.$$

From here on the proof closely the proof from [Yam17b, Section 5.6].

*Game 8.* In this game we change the way $\mathbf{C}$ is chosen. The challenger samples $\mathbf{R}_C \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and sets $\mathbf{C} \coloneqq [\mathbf{A} \mid \mathbf{A}\mathbf{R}_C]$. Everything else remains as in Game 7. By the Leftover Hash Lemma (Lemma 15) we have that the distributions

$$(\mathbf{A}, \mathbf{C}) \qquad \text{and} \qquad (\mathbf{A}, \mathbf{C}')$$

only have negligible statistical difference for $\mathbf{C}' \xleftarrow{\$} \mathbb{Z}_q^{m \times m}$ and therefore

$$\mathsf{E}_8 \geq \mathsf{E}_7 - \mathsf{negl}(\lambda).$$

Before we proceed, we establish some handy observations. Note that for any $\mathsf{id} \in \mathcal{Q}^*$ and $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F, \mathsf{id})$ we have that

$$\|\mathbf{R}_C + \mathbf{R}_{\mathsf{id}}\|_\infty \leq \|\mathbf{R}_C\|_\infty + \|\mathbf{R}_{\mathsf{id}}\|_\infty \leq 1 + \beta \tag{5.15}$$

holds because $\mathbf{R}_C \in \{-1, 1\}^{m \times m}$ and we chose $\mathcal{F}$ to be a correct $(\beta, \gamma, \delta)$-balanced programmable hash function. Also, recall that it follows from the correctness of $\mathcal{F}$ and the abort conditions we established throughout Games 2 to 5 that

$$\mathbf{C} + \mathbf{B}_{\mathsf{id}} = \begin{cases} \mathbf{A}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}}) & \text{if } \mathsf{id} = \mathsf{id}^* \text{ and} \\ \mathbf{A}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}}) + \mathbf{H}_{\mathsf{id}}\mathbf{G} & \text{if } \mathsf{id} \in \mathcal{Q}^* \setminus \{\mathsf{id}^*\}, \end{cases} \tag{5.16}$$

where $\mathbf{H}_{\mathsf{id}} \in \mathsf{GL}(\mathbb{Z}_q, n)$ for all $\mathsf{id} \in \mathcal{Q}^* \setminus \{\mathsf{id}^*\}$.

*Game 9.* This game is identical to the previous game except that the challenger samples $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ instead of generating it with a trapdoor. This makes only a negligible difference by Item 1 in Lemma 26, which states that the statistical distance between $\mathbf{A}$ and a matrix $\mathbf{A}'$ generated using $\mathsf{GenTrap}$ is at most $2^{-n}$.

Further, for each query $\mathsf{id}$ to the $\mathsf{KeyGen}$-oracle the challenger sets $(\mathbf{R}_{\mathsf{id}}, \mathbf{H}_{\mathsf{id}}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F, \mathsf{id})$. Observe that we then have that

$$[\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}}] = [\mathbf{A} \mid \mathbf{A}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}}) + \mathbf{H}_{\mathsf{id}}\mathbf{G}].$$

The challenger then uses the "gadget matrix" $\mathbf{G}$ to compute

$$\mathsf{usk}_{\mathsf{id}} \coloneqq \mathbf{e} \xleftarrow{\$} [\mathbf{A} \mid \mathbf{A}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}}) + \mathbf{H}_{\mathsf{id}}\mathbf{G}]_\sigma^{-1}(\mathbf{u}).$$

Note that the challenger can sample from this distribution by Lemma 14 because $\mathbf{H}_{\mathsf{id}} \in \mathsf{GL}(\mathbb{Z}_q, n)$ and we chose $\sigma \geq m(\beta + m)\omega(\sqrt{\log(m)}) \geq \|\mathbf{R}_C + \mathbf{R}_{\mathsf{id}}\|_\infty$ in Remark 12.

Summing up, choosing $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ alters the view of the adversary only negligibly and the challenger can still sample user secret keys with the correct distribution by using the "gadget matrix". It thus holds that

$$\mathsf{E}_9 \geq \mathsf{E}_8 - \mathsf{negl}(\lambda).$$

*Game 10.* This game is identical to the previous game except that the challenger creates the challenge ciphertext $\mathsf{ct}^*$ differently. As previously, it computes $(\mathbf{R}_{\mathsf{id}^*}, \mathbf{H}_{\mathsf{id}^*}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F, \mathsf{id}^*)$. Recall that we ensured that $\mathbf{H}_{\mathsf{id}^*} = \mathbf{0}$ holds in Game 4. The challenger then samples $b, \kappa_0, \kappa_1 \xleftarrow{\$} \{0,1\}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $x_0 \xleftarrow{\$} D_{\mathbb{Z},\alpha q}$, $\bar{\mathbf{x}}_1 \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha q}$ and sets $w_0 \coloneqq \mathbf{s}^\mathsf{T}\mathbf{u} + x_0$ and $\mathbf{w}_1^\mathsf{T} \coloneqq \mathbf{s}^\mathsf{T}\mathbf{A} + \bar{x}_1^\mathsf{T}$. Then it defines the challenge ciphertext $\mathsf{ct}^* \coloneqq (c_0, \mathbf{c}_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2m}$ as

$$c_0 \coloneqq w_0 + \kappa_1 \cdot \lceil q/2 \rceil, \quad \mathbf{c}_1^\mathsf{T} \coloneqq \mathsf{ReRand}([\mathbf{I}_m \mid \mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}], \mathbf{w}_1, \alpha q, \alpha'/2\alpha), \tag{5.17}$$

where $\mathbf{I}_m$ is the identity matrix of dimension $m$ and $\mathsf{ReRand}$ is the algorithm from Lemma 25. The challenger outputs $(\mathsf{ct}^* = (c_0, \mathbf{c}_T), \kappa_b)$.

By Lemma 25, this alters the view of the adversary only negligibly. More precisely, define the matrix $\mathbf{V}$ and the vector $\mathbf{b}^\mathsf{T}$ as follows:

$$\mathbf{V} \coloneqq [\mathbf{I}_m \mid \mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}] \qquad \mathbf{x} \coloneqq \bar{\mathbf{x}}_1 \qquad \text{and} \qquad \mathbf{b}^\mathsf{T} \coloneqq \mathbf{s}^\mathsf{T}\mathbf{A}.$$

By applying Lemma 25 we then obtain that the distribution of $\mathbf{c}_1^\mathsf{T}$ as computed in Equation (5.17) above has only a negligible statistical distance to

$$\begin{aligned} \mathbf{c}_1^\mathsf{T} &= \mathbf{s}^\mathsf{T}\mathbf{A}^\mathsf{T}[\mathbf{I}_m \mid \mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}] + [\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}] && (5.18) \\ &= \mathbf{s}^\mathsf{T}[\mathbf{A} \mid \mathbf{A}\mathbf{R}_C + \mathbf{A}\mathbf{R}_{\mathsf{id}^*}] + [\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}] \\ &= \mathbf{s}^\mathsf{T}[\mathbf{A} \mid \mathbf{C} + \mathbf{B}_{\mathsf{id}^*}] + [\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}], && (5.19) \end{aligned}$$

where $\mathbf{x}_1, \mathbf{x}_2 \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha' q}$. Note that Equation (5.18) holds because of Equation (5.16) and Equation (5.19) holds because $\mathbf{C} = \mathbf{A}\mathbf{R}_C$ and $\mathbf{B}_{\mathsf{id}^*} = \mathbf{A}\mathbf{R}_{\mathsf{id}^*}$. We can apply Lemma 25 because

$$\alpha'/(2\alpha) > \sqrt{2} \cdot m \cdot (\beta + 1) \geq \sqrt{2m}\sqrt{m} \cdot \|\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}\|_\infty \geq \|\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}\|_2 ,$$

where the second inequality follows from Equation (5.15) with $\mathsf{id} = \mathsf{id}^*$ and the third from the general inequality between the two norms. Therefore, we conclude that

$$\mathsf{E}_{10} \geq \mathsf{E}_9 - \mathsf{negl}(\lambda).$$

*Game 11.* In this game, the generation procedure for the challenge ciphertext $\mathsf{ct}^*$ is modified again. First, the challenger picks $v_0 \xleftarrow{\$} \mathbb{Z}_q$, $\mathbf{v}_1 \xleftarrow{\$} \mathbb{Z}_q^m$, $x_0 \xleftarrow{\$} D_{\mathbb{Z},\alpha q}$, and vectors $\mathbf{x}_1 \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha' q}$. It then uses $\mathsf{TrapEval}$ to compute $(\mathbf{R}_{\mathsf{id}^*}, \mathbf{H}_{\mathsf{id}^*}) \coloneqq \mathsf{TrapEval}(\mathsf{td}, F, \mathsf{id}^*)$. Next, it samples $b, \kappa_0, \kappa_1 \xleftarrow{\$} \{0,1\}$ as in the $\mathsf{IND\text{-}ID\text{-}CPA}$ security experiment and sets the challenge ciphertext as in Equation (5.17) of Game 10, but with $w_0 \coloneqq v_0 + x_0$ and $\mathbf{w}_1 \coloneqq \mathbf{v}_1 + \bar{\mathbf{x}}_1$. Specifically, the challenger sets

$$c_0 \coloneqq v_0 + x_0 + \kappa_1 \cdot \lceil q/2 \rceil$$

and

$$\mathbf{c}_1^\mathsf{T} \coloneqq \mathsf{ReRand}([\mathbf{I}_m \mid \mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}], \mathbf{v} + \bar{\mathbf{x}}_1, \alpha q, \alpha'/2\alpha).$$

We then claim that

$$\mathsf{E}_{11} \geq \mathsf{E}_{10} - \mathsf{negl}(\lambda)$$

holds under the $\mathsf{dLWE}_{n,m+1,q,\alpha}$ assumption. We prove the claim by constructing an adversary $\mathcal{B}$ against $\mathsf{dLWE}_{n,m+1,q,\alpha}$ from any algorithm $\mathcal{A}$ that distinguishes between Game 10 and Game 11 with non-negligible advantage.

Description of $\mathcal{B}$. The algorithm receives a $\mathsf{dLWE}_{n,m+1,q,\alpha}$ instance $(\mathbf{A}', \mathbf{w}' = \mathbf{v}' + \bar{x}) \in \mathbb{Z}_q^{n\times(m+1)} \times \mathbb{Z}_q^{m+1}$ as input, where $\bar{\mathbf{x}} \xleftarrow{\$} D_{\mathbb{Z}^{m+1},\alpha q}$. The algorithm then tries to distinguish whether it holds that $\mathbf{v}'^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{A}'$ for $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ or if it holds that $\mathbf{v}' \xleftarrow{\$} \mathbb{Z}_q^{m+1}$ as follows.

**Generating $\mathsf{mpk}$:** Denote the first column of $\mathbf{A}'$ by $\mathbf{u} \in \mathbb{Z}_q^n$ and denote the last $m$ columns of $\mathbf{A}'$ by $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. Also, denote the first coefficient of $\mathbf{w}' \in \mathbb{Z}_q^{m+1}$ by $w_0 \in \mathbb{Z}_q$ and let the last $m$ coefficients of $\mathbf{w}'$ be $\mathbf{w}_1 \in \mathbb{Z}_q^m$. Then the algorithm $\mathcal{B}$ sets the master public key $\mathsf{mpk}$ exactly as both in Game 10 and Game 11.

**Answering queries and creating $\mathsf{ct}^*$:** Just as since Game 2, $\mathcal{B}$ aborts and outputs a random bit if $\mathcal{A}$ queries $\mathsf{id}^{(i)}$ to $\mathsf{KeyGen}$-oracle or chooses a challenge identity $\mathsf{id}^*$ such that

$$\mathbf{H}_{\mathsf{id}^{(i)}} = \mathbf{0} \qquad \text{or} \qquad \mathbf{H}_{\mathsf{id}^*} \neq \mathbf{0}.$$

Under these conditions, $\mathcal{A}$ can answer key extraction queries as described in Game 9 and in particular without knowledge of a trapdoor for $\mathbf{A}$. Furthermore, it can construct $\mathsf{ct}^*$ exactly as in Equation (5.17) by sampling $b, \kappa_0, \kappa_1 \xleftarrow{\$} \{0,1\}$ and setting $\mathsf{ct}^* \coloneqq (c_0, \mathbf{c}_1^\mathsf{T})$. It then outputs $((c_0, \mathbf{c}_1^\mathsf{T}), \kappa_b)$ to $\mathcal{A}$. Note that the secret randomness used to generate $w_0$ and $\mathbf{w}_1$ (namely, $\mathbf{s}$ and $\bar{\mathbf{x}}$) is not necessary for this computation.

**Solving dLWE:** Once $\mathcal{A}$ outputs its guess $b'$, our algorithm $\mathcal{B}$ checks whether $b' = b$ and if so outputs 1 and 0 otherwise.

Analysis of $\mathcal{B}$'s success probability. Observe that if $(\mathbf{A}', \mathbf{w}')$ is a valid LWE sample with $\mathbf{v}'^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{A}'$, then $\mathcal{A}$'s view corresponds to Game 10. Otherwise, that is if $\mathbf{v}' \xleftarrow{\$} \mathbb{Z}_q^{m+1}$, then $\mathcal{A}$'s view corresponds Game 11. Hence, we have that the advantage of $\mathcal{B}$ in solving $\mathsf{dLWE}_{n,m+1,q,\alpha}$ is $(\Pr[1 \xleftarrow{\$} \mathsf{G}_{11}] - \Pr[1 \xleftarrow{\$} \mathsf{G}_{10}])$ and we thus conclude that

$$\mathsf{E}_{11} \geq \mathsf{E}_{10} - \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda).$$

*Game 12.* We further alter the challenge ciphertext in this game. The challenger now samples $v_0 \xleftarrow{\$} \mathbb{Z}_q, \mathbf{v}_1 \xleftarrow{\$} \mathbb{Z}_q^m, x_0 \xleftarrow{\$} D_{\mathbb{Z},\alpha q}$ and $\mathbf{x}_1, \mathbf{x}_2 \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha' q}$ and computes $\mathbf{R}_{\mathsf{id}^*}$ using $\mathsf{TrapEval}$. Then the challenger samples $b, \kappa_0, \kappa_1 \xleftarrow{\$} \{0,1\}$ and sets the challenge ciphertext as

$$c_0 \coloneqq v_0 + x_0 + \kappa_1 \cdot \lceil q/2 \rceil, \qquad \mathbf{c}_1^\mathsf{T} \coloneqq \left[\mathbf{v}_1^\mathsf{T} \mid \mathbf{v}_1^\mathsf{T}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*})\right] + \left[\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2^\mathsf{T}\right]. \tag{5.20}$$

Note that the difference between this game and the previous one is that the challenger samples both $\mathbf{x}_1$ and $\mathbf{x}_2$ in contrast to Game 11 where the challenger samples only $\bar{\mathbf{x}_1}$. This can be seen as reverting the change from Game 9 to Game 10. Like that previous change, also this one alters the view of the adversary only negligibly. We can verify this by setting $\mathbf{V} \coloneqq [\mathbf{I}_m \mid \mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}], \bar{\mathbf{x}}_1 \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha' q}$ and $\mathbf{v}_1 \xleftarrow{\$} \mathbb{Z}_q^m$ (as in Game 11). It then follows that

$$\mathsf{ReRand}(\mathbf{V}, \mathbf{v}_1 + \bar{\mathbf{x}}_1, \alpha q, \alpha'/2\alpha),$$

which is how $\mathbf{c}_1^\mathsf{T}$ is computed in Game 11, is statistically negligibly close to $\mathbf{c}_1^\mathsf{T}$ in Equation (5.20) above by Lemma 25. This lemma is applicable because we chose the parameters in Remark 12 such that

$$\alpha'/(2\alpha) > \sqrt{2} \cdot m \cdot (\beta + 1) \geq \sqrt{2m}\sqrt{m} \cdot \|\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}\|_\infty \geq \|\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*}\|_2,$$

where the second inequality follows from Equation (5.15) with $\mathsf{id} = \mathsf{id}^*$ and the third from the general inequality between the two norms. We thus have that

$$\mathsf{E}_{12} \geq \mathsf{E}_{11} - \mathsf{negl}(\lambda).$$

*Game 13.* In this game, we change the challenge ciphertext such that it contains no information about $\kappa_1$ anymore. Formally, we set $\mathsf{ct}^* \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q^{2m}$. We then have that

$$\mathsf{E}_{13} = 0.$$

Thus, it is only left to show that $\mathsf{E}_{13} \geq \mathsf{E}_{12} - \mathsf{negl}(\lambda)$. Both games differ only in the generation of the challenge ciphertext. We thus focus on this part. First, we observe that $c_0$ is already uniformly random in Game 12 because we there already sample $v_0 \xleftarrow{\$} \mathbb{Z}_q$. We can therefore limit our attention to showing that the distribution of $\mathbf{c}_1$ is negligibly close to the uniform distribution over $\mathbb{Z}_q^{2m}$. Recall that

$$\mathbf{c}_1 = \left[\mathbf{v}_1^\mathsf{T} \mid \mathbf{v}_1^\mathsf{T}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*})\right] + \left[\mathbf{x}_1^\mathsf{T} \mid \mathbf{x}_2\mathsf{T}\right].$$

We begin by observing that for $\mathbf{A}, \mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{R}_C \xleftarrow{\$} \{-1,1\}^{m\times m}, \mathbf{v}_1, \mathbf{v}_1'^\mathsf{T} \xleftarrow{\$} \mathbb{Z}_q^m$ the distributions

$$\left(\mathbf{A}, \mathbf{A}\mathbf{R}_C, \mathbf{v}_1^\mathsf{T}, \mathbf{v}_1^\mathsf{T}\mathbf{R}_C\right) \approx \left(\mathbf{A}, \mathbf{A}', \mathbf{v}_1^\mathsf{T}, \mathbf{v}_1'^\mathsf{T}\right) \approx \left(\mathbf{A}, \mathbf{A}\mathbf{R}_C, \mathbf{v}_1^\mathsf{T}, \mathbf{v}_1'^\mathsf{T}\right), \tag{5.21}$$

are negligibly close. We note that the first two distributions are negligibly close by the leftover hash lemma (Lemma 15) for $\left[\mathbf{A}^\mathsf{T} \mid \mathbf{v}_1\right]^\mathsf{T} \in \mathbb{Z}_q^{(n+1)\times m}$ and $\mathbf{R}_C$. We obtain the statistical closeness for the second and the third distribution by applying the leftover hash lemma for $\mathbf{A}$ and $\mathbf{R}_C$.

Using Equation (5.21), we conclude that for $\mathbf{A}, \mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{R}_C \xleftarrow{\$} \{-1,1\}^{m\times m}$ and $\mathbf{v}_1, \mathbf{v}_1'^\mathsf{T} \xleftarrow{\$} \mathbb{Z}_q^m$ it holds that

$$\begin{aligned}\left(\mathbf{A}, \mathbf{A}\mathbf{R}_C, \mathbf{c}_1^\mathsf{T}, \mathbf{v}_1'^\mathsf{T}\right) &\approx \left(\mathbf{A}, \mathbf{A}\mathbf{R}_C, \mathbf{v}_1^\mathsf{T} + \mathbf{x}_1^\mathsf{T}, \mathbf{v}_1'^\mathsf{T} + \mathbf{v}_1^\mathsf{T}\mathbf{R}_{\mathsf{id}^*} + \mathbf{x}_2^\mathsf{T}\right) \\ &\approx \left(\mathbf{A}, \mathbf{A}\mathbf{R}_C, \mathbf{v}_1^\mathsf{T} + \mathbf{x}_1^\mathsf{T}, \mathbf{v}_1^\mathsf{T}(\mathbf{R}_C + \mathbf{R}_{\mathsf{id}^*} + \mathbf{x}_2^\mathsf{T})\right)\end{aligned}$$

are negligibly close. The first and second distribution are negligibly close because

1. $\mathbf{x}_1, \mathbf{x}_2$ are chosen independently at random from the other variables and
2. $\mathbf{R}_{\mathsf{id}^*}$ is computed from the trapdoors $(\mathbf{R}_i)_{0\leq i\leq \ell}$, which are also chosen independently at random from the other variables.

Finally, it immediately follows from Equation (5.21) that the second and the third distributions are negligibly close by. We thus conclude that

$$\mathsf{E}_{13} \geq \mathsf{E}_{12} - \mathsf{negl}(\lambda).$$

Analysis. From all the games above, we can now conclude that

$$\begin{aligned}0 = \mathsf{E}_{13} &\geq \mathsf{E}_{11} - \mathsf{negl}(\lambda) \geq \mathsf{E}_{10} - \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda) - \mathsf{negl}(\lambda) \\ &\geq \mathsf{E}_4 - \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda) - \mathsf{negl}(\lambda) \\ &\geq \mathsf{E}_3 \cdot \Pr\left[\neg\mathsf{badChal}_{\mathsf{phf}}\right] - \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda) - \mathsf{negl}(\lambda) \\ &\geq (\mathsf{E}_2 - \Pr\left[\mathsf{coll}_{\mathsf{phf}}\right]) \cdot \Pr\left[\neg\mathsf{badChal}_{\mathsf{phf}}\right] - \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda) - \mathsf{negl}(\lambda) \\ &= \delta(t,\varepsilon) - \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda) - \mathsf{negl}(\lambda).\end{aligned}$$

This is equivalent to

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{dLWE}_{n,m+1,q,\alpha}}(\lambda) \geq \delta(t,\varepsilon) - \mathsf{negl}(\lambda),$$

which concludes the proof of Theorem 13. □

## 5.8 Conclusion and Open Questions

Concluding the chapter, we introduced balanced PHFs for lattices and demonstrated how this extension of PHFs for lattices enables the construction of IB-KEMs from lattices with security in the standard model. Moreover, we presented three constructions of balanced PHFs. Our first balanced PHF $\mathcal{F}_{\mathsf{BLK}}$ is based on blockwise partitioning, which we previously introduced in Chapter 3. This construction shows how blockwise partitioning can be used in the context of lattices.

However, the downside of blockwise partitioning is that it relies on the non-standard assumption of weak near-collision resistant hash functions. We addressed this shortcoming by introducing the weaker notion of ECR hash functions and presenting a construction from the eSIS assumption, a variant of the SIS assumption with a concrete bound on the running time of algorithms. We note that exponential-collision resistance is a stronger assumption than truncation-collision resistance and weak near-collision resistance, and thus every TCR or wNCR hash function is also an ECR hash function. Thus, the advantage of ECR hash functions is that we are able to provide an explicit construction from a well-studied computational problem.

We then demonstrated that ECR hash functions suffice to construct balanced PHFs by presenting two balanced PHFs from ECR hash functions. The first balanced PHFs $\mathcal{F}_{\mathsf{AS}}$ has a descriptions size of only $\mathcal{O}(\lambda^2 \log(\lambda))$ many $\mathbb{Z}_q$-elements. It is based on a technique by Alperin-Sheriff [Alp15] and thus inherits a super-polynomial LWE parameter from it. We thus presented $\mathcal{F}_{\mathsf{SN}}$, which has a slightly larger description size of $\mathcal{O}(\sqrt{\log \lambda} \cdot \lambda^2 \log(\lambda))$ many $\mathbb{Z}_q$-elements but has a polynomially bounded LWE parameter. Unfortunately, both balanced PHFs from ECR hash functions suffer from a super-polynomial but, sub-quasi-polynomial reduction loss of $\lambda^{\mathcal{O}(\log\log(\lambda))}$ that originates from the fact that our ECR hash function produces outputs in $\mathbb{Z}_q^n$, where $n$ can be chosen freely, and $q$ is polynomially bounded in $n$. A construction of ECR hash functions for which $q$ is a constant would resolve this issue.

**Research Question 6.** *Are there ECR hash functions whose security can be reduced to the hardness of well-studied computational problems and produce outputs in $\mathbb{Z}_q^n$ for a constant $q$?*

However, even given such ECR hash functions, our most efficient balanced PHF $\mathcal{F}_{\mathsf{AS}}$ would still result in an IB-KEM, which requires a super-polynomial LWE parameter for security. This leads to the open question already asked by Peikert in [Pei16, Question 9].

**Research Question 7.** *Are there standard-model, adaptively-secure lattice-based IBE/IB-KEM schemes that have comparable efficiency and concrete security to the existing selectively-secure ones?*

# Bibliography


[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

[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, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_28.

[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, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-14623-7_6.

[ACD+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 351–367, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-98113-0_19.

[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. doi:10.1007/s00145-013-9153-x.

[ADH+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 717–746, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17656-3_25.

[AFL12] Michel Abdalla, Dario Fiore, and Vadim Lyubashevsky. From selective to full security: Semi-generic transformations in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis,

editors, *PKC 2012*, volume 7293 of *LNCS*, pages 316–333, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-30057-8_19.

[AFL16] Daniel Apon, Xiong Fan, and Feng-Hao Liu. Compact identity based encryption from LWE. Cryptology ePrint Archive, Report 2016/125, 2016. https://eprint.iacr.org/2016/125.

[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.

[AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-29011-4_34.

[AGO11] Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-25385-0_34.

[AGVW17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 297–322, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70694-8_11.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108, Philadephia, PA, USA, May 22–24, 1996. ACM Press. doi:10.1145/237814.237838.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-56614-6_4.

[Alp15] Jacob Alperin-Sheriff. Short signatures with short public keys from homomorphic trapdoor functions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 236–255, Gaithersburg, MD, USA,

March 30 – April 1, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46447-2_11.

[AMN+19] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively single-key secure constrained PRFs for $NC^1$. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 223–253, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17259-6_8.

[AMNR18] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, and Ling Ren. Dfinity consensus, explored. Cryptology ePrint Archive, Report 2018/1153, 2018. https://eprint.iacr.org/2018/1153.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015. URL: https://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml.

[ASM07] Man Ho Au, Willy Susilo, and Yi Mu. Practical compact e-cash. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP 07*, volume 4586 of *LNCS*, pages 431–445, Townsville, Australia, July 2–4, 2007. Springer, Heidelberg, Germany.

[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, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_14.

[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, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-28628-8_27.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. doi:10.1007/11426639_26.

[BC04] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 290–305, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-28628-8_18.

[BCJ+05] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and reduced SHA-1. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 36–57, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. doi:10.1007/11426639_3.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131, Palo Alto, CA, USA, August 12–14, 2009. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-03298-1_9.

[BDE+21] Maxime Buser, Rafael Dowsley, Muhammed F. Esgin, Shabnam Kasra Kermanshahi, Veronika Kuchta, Joseph K. Liu, Raphael Phan, and Zhenfei Zhang. Post-quantum verifiable random function from symmetric primitives in pos blockchain. Cryptology ePrint Archive, Report 2021/302, 2021. https://eprint.iacr.org/2021/302.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM. doi:10.1137/1.9781611974331.ch2.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-78967-3_11.

[Ber45] Joseph Bertrand. Mémoire sur le nombre de valeurs que peut prendre une fonction: quand on y permute les lettres qu'elle renferme. *Journal de l'École Royale Polytechnique*, 18:123–140, 1845.

[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, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_13.

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[BFMS08] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. *Journal of Cryptology*, 21(2):178–199, April 2008. doi:10.1007/s00145-007-9000-z.

[BGK+18] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930, Toronto, ON, Canada, October 15–19, 2018. ACM Press. doi:10.1145/3243734.3243848.

[BGLS19] Shi Bai, Steven D. Galbraith, Liangze Li, and Daniel Sheffield. Improved combinatorial algorithms for the inhomogeneous short integer solution problem. *Journal of Cryptology*, 32(1):35–83, January 2019. doi:10.1007/s00145-018-9304-1.

[BHJ+13] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, Jae Hong Seo, and Christoph Striecks. Practical signatures from standard assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 461–485, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-38348-9_28.

[BHJ+15] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, and Christoph Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, 28(1):176–208, January 2015. doi:10.1007/s00145-014-9183-z.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press. doi:10.1145/2382196.2382279.

[Bit17a] 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, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70503-3_19.

[Bit17b] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. Cryptology ePrint Archive, Report 2017/018, 2017. https://eprint.iacr.org/2017/018.

[Bit20] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. *Journal of Cryptology*, 33(2):459–493, April 2020. doi:10.1007/s00145-019-09331-1.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume

9666 of *LNCS*, pages 273–304, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_10.

[BKKP15] Olivier Blazy, Saqib A. Kakvi, Eike Kiltz, and Jiaxin Pan. Tightly-secure signatures from chameleon hash functions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 256–279, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46447-2_12.

[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, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_23.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd ACM STOC*, pages 435–440, Portland, OR, USA, May 21–23, 2000. ACM Press. doi:10.1145/335305.335355.

[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, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53890-6_14.

[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, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. doi:10.1145/2488608.2488680.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004. doi:10.1007/s00145-004-0314-9.

[BMR10a] Dan Boneh, Hart Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. Cryptology ePrint Archive, Report 2010/442, 2010. https://eprint.iacr.org/2010/442.

[BMR10b] 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, Chicago, Illinois, USA, October 4–8, 2010. ACM Press. doi:10.1145/1866307.1866323.

[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, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13013-7_29.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. doi:10.1145/168588.168596.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany. doi:10.1007/3-540-68339-9_34.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. doi:10.1007/11761679_25.

[BR09a] 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, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-01001-9_24.

[BR09b] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' IBE scheme. Cryptology ePrint Archive, Report 2009/084, 2009. https://eprint.iacr.org/2009/084.

[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, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53015-3_13.

[CD96] Ronald Cramer and Ivan Damgård. New generation of secure and practical RSA-based signatures. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 173–185, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany. doi:10.1007/3-540-68697-5_14.

[CFN15] Dario Catalano, Dario Fiore, and Luca Nizzardo. Programmable hash functions go private: Constructions and applications to (homomorphic) signatures with shorter public keys. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 254–274, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-48000-7_13.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press. doi:10.1145/276698.276741.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. doi:10.1145/1008731.1008734.

[Che52] Pafnuty Lvovich Chebyshev. Mémoire sur les nombres premiers. *Journal de mathématiques pures et appliquées*, 17:366–390, 1852.

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. doi:10.1007/11761679_1.

[Che10] Jung Hee Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, July 2010. doi:10.1007/s00145-009-9047-0.

[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, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_27.

[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. doi:10.1007/s00145-011-9105-2.

[CHZ16] Yu Chen, Qiong Huang, and Zongyang Zhang. Sakai-ohgishi-kasahara identity-based non-interactive key exchange revisited and more. *Int. J. Inf. Sec.*, 15(1):15–33, 2016. doi:10.1007/s10207-015-0274-0.

[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, Cirencester, UK, December 17–19, 2001. Springer, Heidelberg, Germany.

[Cor01] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. Cryptology ePrint Archive, Report 2001/062, 2001. https://eprint.iacr.org/2001/062.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. doi:10.1007/3-540-46035-7_18.

[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, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_25.

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34805-0_39.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_3.

[DKN+20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 559–589, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56784-2_19.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_19.

[DMS99] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 475–485. IEEE Computer Society, New York, USA, 1999. doi:10.1109/SFFCS.1999.814620.

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 125–145, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78381-9_5.

[EKS+20] Muhammed F. Esgin, Veronika Kuchta, Amin Sakzad, Ron Steinfeld, Zhenfei Zhang, Shifeng Sun, and Shumo Chu. Practical post-quantum few-time verifiable random function with applications to algorand. Cryptology ePrint Archive, Report 2020/1222, 2020. https://eprint.iacr.org/2020/1222.

[FHJ20] Marc Fischlin, Patrick Harasser, and Christian Janson. Signatures from sequential-OR proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 212–244, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45727-3_8.

[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, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_28.

[FLR+10] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 303–320, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-17373-8_18.

[GCS+17] Fuchun Guo, Rongmao Chen, Willy Susilo, Jianchang Lai, Guomin Yang, and Yi Mu. Optimal security reductions for unique signatures:

Bypassing impossibilities with a counterexample. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 517–547, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-63715-0_18.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017. doi:10.1145/3132747.3132757.

[Gil52] Edgar Nelson Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31:504–522, May 1952. doi:10.1002/j.1538-7305.1952.tb01393.x.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GNP+15] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: Provably preventing DNSSEC zone enumeration. In *NDSS 2015*, San Diego, CA, USA, February 8–11, 2015. The Internet Society.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008. doi:10.1017/CBO9780511804106.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008. doi:10.1016/j.dam.2007.12.010.

[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, Victoria, BC, Canada, May 17–20, 2008. ACM Press. doi:10.1145/1374376.1374407.

[Gra07] Markus Grassl. Bounds on the minimum distance of linear codes and quantum codes. Online available at `http://www.codetables.de`, 2007. Accessed on 2021-07-15.

[GRPV21] Sharon Goldberg, Leonid Reyzin, Dimitrios Papadopoulos, and Jan Včelák. Verifiable Random Functions (VRFs). Internet-Draft draft-irtf-cfrg-vrf-09, Internet Engineering Task Force, May 2021. Work in Progress. URL: `https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-vrf-09`.

[HHK+21] Yuncong Hu, Kian Hooshmand, Harika Kalidhindi, Seung Jin Yang, and Raluca Ada Popa. Merkle$^2$: A low-latency transparency log system. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 285–303, 2021. `doi:10.1109/SP40001.2021.00088`.

[HJ12] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-32009-5_35`.

[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, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. `doi:10.1007/978-3-662-49096-9_14`.

[HJK11] Dennis Hofheinz, Tibor Jager, and Eike Kiltz. Short signatures from weaker assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 647–666, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-25385-0_35`.

[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, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-30057-8_5`.

[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, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. `doi:10.1007/978-3-540-85174-5_2`.

[HK12] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, 25(3):484–527, July 2012. doi:10.1007/s00145-011-9102-5.

[HP03] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003. doi:10.1017/CBO9780511807077.

[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, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_33.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442, Victoria, BC, Canada, May 17–20, 2008. ACM Press. doi:10.1145/1374376.1374438.

[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, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_5.

[JK18] Tibor Jager and Rafael Kurek. Short digital signatures and ID-KEMs via truncation collision resistance. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 221–250, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03329-3_8.

[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, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75245-3_22.

[JKP18] Tibor Jager, Rafael Kurek, and Jiaxin Pan. Simple and more efficient PRFs with tight security from LWE and matrix-DDH. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 490–518, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03332-3_18.

[JN19a] 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, Waterloo, ON, Canada, August 12–16, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-38471-5_13.

[JN19b] Tibor Jager and David Niehues. On the real-world instantiability of admissible hash functions and efficient verifiable random functions. Cryptology ePrint Archive, Report 2019/1335, 2019. https://eprint.iacr.org/2019/1335.

[JS04] Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 590–608, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_35.

[Jus72] Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Trans. Information Theory*, 18(5):652–656, 1972. doi:10.1109/TIT.1972.1054893.

[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, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70700-6_4.

[Ker83a] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, IX:5–38, January 1883. A copy of the paper is available at https://www.petitcolas.net/kerckhoffs/index.html.

[Ker83b] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, IX:161–191, Febuary 1883. A copy of the paper is available at https://www.petitcolas.net/kerckhoffs/index.html.

[KK12] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 537–553, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-29011-4_32.

[KK18] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. *Journal of Cryptology*, 31(1):276–306, January 2018. doi:10.1007/s00145-017-9257-9.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014. URL: https:

//www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570276.

[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, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17259-6_14.

[KONT20] Kaisei Kajita, Kazuto Ogawa, Koji Nuida, and Tsuyoshi Takagi. Short lattice signatures in the standard model with efficient tag generation. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *Provable and Practical Security - 14th International Conference, ProvSec 2020, Singapore, November 29 - December 1, 2020, Proceedings*, volume 12505 of *Lecture Notes in Computer Science*, pages 85–102. Springer, Heidelberg, Germany, 2020. doi:10.1007/978-3-030-62576-4_5.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS 2000*, San Diego, CA, USA, February 2–4, 2000. The Internet Society.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164, Washington, DC, USA, October 27–30, 2003. ACM Press. doi:10.1145/948109.948132.

[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, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53890-6_23.

[Ler21] Antonin Leroux. Proofs of isogeny knowledge and application to post-quantum one-time verifiable random function. Cryptology ePrint Archive, Report 2021/744, 2021. https://eprint.iacr.org/2021/744.

[Lis05] Moses Liskov. Updatable zero-knowledge databases. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 174–198, Chennai, India, December 4–8, 2005. Springer, Heidelberg, Germany. doi:10.1007/11593447_10.

[LPC85] Edwin T Layton, Roger Pineau, and John Costello. *"And I was There": Pearl Harbor and Midway – breaking the Secrets*. William Morrow & Co, New York, USA, December 1985.

[LST18] Benoît Libert, Damien Stehlé, and Radu Titiu. Adaptively secure distributed PRFs from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 391–421, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03810-6_15.

[LV20] Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to PPAD-hardness and VDFs. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 632–651, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56877-1_22.

[LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is difficult. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 58–76, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_4.

[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, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. doi:10.1007/3-540-45708-9_38.

[MBB+15] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing key transparency to end users. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 383–398, Washington, DC, USA, August 12–14, 2015. USENIX Association.

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34805-0_21.

[MGZ+18] Antonio Marcedone, Cesar Ghali, Daniel Ziegler, Gary Belvin, and Ismail Khoffi. Google key transparency. Online available at https://github.com/google/keytransparency, 2018. Accessed on 2021-08-09, Authors are listed according to https://github.com/google/keytransparency/blob/master/CONTRIBUTORS, the copyright is held by Antonio Marcedone, Google Inc. and Yahoo! Inc. according to https://github.com/google/keytransparency/blob/master/AUTHORS.

[MM13] Jaban Meher and M. Ram Murty. Ramanujan's proof of bertrand's postulate. *Am. Math. Mon.*, 120(7):650–653, 2013. doi:10.4169/amer.math.monthly.120.07.650.

[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, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-29011-4_41.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_2.

[MP18] Andrew Morgan and Rafael Pass. On the security loss of unique signatures. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 507–536, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03807-6_19.

[MPS20] Andrew Morgan, Rafael Pass, and Elaine Shi. On the adaptive security of MACs and PRFs. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 724–753, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64837-4_24.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 542–565, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_32.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 149–163, San Jose, CA, USA, February 18–22, 2002. Springer, Heidelberg, Germany. doi:10.1007/3-540-45760-7_11.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. doi:10.1137/S0097539705447360.

[MRRW77] Robert J. McEliece, Eugene R. Rodemich, Howard Rumsey, Jr., and Lloyd R. Welch. New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *IEEE Trans. Information Theory*, 23(2):157–166, 1977. doi:10.1109/TIT.1977.1055688.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press. doi:10.1109/SFFCS.1999.814584.

[MS98] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*. North Holland mathematical library. Amsterdam [u.a.] : North Holland, 10. impr. edition, 1998.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 5th edition, 1996. URL: https://cacr.uwaterloo.ca/hac/, doi:10.1201/9781439821916.

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 820–849, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_31.

[Nie21] David Niehues. Verifiable random functions with optimal tightness. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 61–91, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75248-4_3.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. doi:10.1109/SFCS.1997.646134.

[oST15a] National Institute of Standards and Technology. Fips pub 180–4: Secure hash standard, August 2015. doi:10.6028/NIST.FIPS.180-4.

[oST15b] National Institute of Standards and Technology. Fips pub 202: Sha-3 standard: Permutation-based hash and extendable-output functions, August 2015. doi:10.6028/NIST.FIPS.202.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536461.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016. doi:10.1561/0400000074.

[Pet11] Fabien A. P. Petitcolas. Kerckhoffs' principle. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed*, page 675. Springer, Heidelberg, Germany, 2011. doi:10.1007/978-1-4419-5906-5_487.

[PS14] Inna Polak and Adi Shamir. Using random error correcting codes in near-collision attacks on generic hash-functions. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 219–236, New Delhi, India, December 14–17, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-13039-2_13.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-85174-5_31.

[PW88] William Wesley Peterson and Edward J. Weldon. *Error-correcting codes*. MIT Press, Cambridge, Massachusetts, USA, 2. ed., 9. print. edition, 1988.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. doi:10.1145/1060590.1060603.

[Riv90] Ronald L. Rivest. Cryptography. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 717–755. Elsevier and MIT Press, 1990.

[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, Naples, Italy, September 30 – October 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-00434-7_22.

[SAK+01] Kenneth W. Shum, Ilia Aleshnikov, P. Vijay Kumar, Henning Stichtenoth, and Vinay Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the gilbert-varshamov bound. *IEEE Trans. Information Theory*, 47(6):2225–2241, 2001. doi:10.1109/18.945244.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

[Sch11] Sven Schäge. Tight proofs for signature schemes without random oracles. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 189–206, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-20465-4_12.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994. doi:10.1007/BF01581144.

[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, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[Sho00] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 275–288, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-45539-6_19.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332.

[SS96] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Trans. Information Theory*, 42(6):1710–1722, 1996. doi:10.1109/18.556667.

[Tur37] Alan Mathison Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, January 1937. doi:10.1112/plms/s2-42.1.230.

[Var57] Rom Rubenovich Varshamov. Estimate of the number of signals in error correcting codes. *Dokl. Acad. Nauk SSSR*, 117(5):739–741, 1957.

[Var97] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Information Theory*, 43(6):1757–1766, 1997. doi:10.1109/18.641542.

[VGP+18] Jan Včelák, Sharon Goldberg, Dimitrios Papadopoulos, Shumon Huque, and David C. Lawrence. NSEC5, DNSSEC Authenticated Denial of Existence. Internet-Draft draft-vcelak-nsec5-08, Internet Engineering Task Force, December 2018. Work in Progress. URL: https://datatracker.ietf.org/doc/html/draft-vcelak-nsec5-08.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. doi:10.1007/11426639_7.

[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, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_2.

[Yam17a] 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, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-63697-9_6.

[Yam17b] Shota Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. Cryptology ePrint Archive, Report 2017/096, 2017. https://eprint.iacr.org/2017/096.

[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, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53015-3_11.

[Zém01] Gilles Zémor. On expander codes. *IEEE Trans. Information Theory*, 47(2):835–837, 2001. doi:10.1109/18.910593.

[Zha16] Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53018-4_18.

[Zha19a] Mark Zhandry. The magic of ELFs. *Journal of Cryptology*, 32(3):825–866, July 2019. doi:10.1007/s00145-018-9289-9.

[Zha19b] Mark Zhandry. On ELFs, deterministic encryption, and correlated-input security. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 3–32, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17659-4_1.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, Heidelberg, Germany, 1979. doi:10.1007/3-540-09519-5_73.

# A The Program Used to find Parameters of BCH Codes

```
import math as m
import logging

logging.basicConfig()
logger = logging.getLogger("find_good_bch_codes")
logger.setLevel(logging.WARNING)


def list_suitable_bch_codes(n, min_dimension,
    required_relative_minimal_bose_distance):
  """Lists parameters of BCH codes that (potentially after
    puncturing) have codewords of length n, dimension at least
    min_dimension and has relative distance of at least
    required_relative_minimal_bose_distance."""
  # the minimal distance required in order to achieve the given
  # relative distance.
  required_min_distance = m.ceil(n *
    required_relative_minimal_bose_distance)

  # compute all 2-cyclotomic cosets mod n.
  cosets = q_cyclotomic_cosets_mod_n(2, n)
  print("Finished computing cyclotomic cosets")

  smallest_n = n + 1
  best_k = 0
  best_d = 0
  best_puncturing = 0

  # consider all possible starting points for sequences for
  # 2-cyclotomic cosets mod n
  for start in range(0, n - 1):
    # initialize the defining set to be empty and i to be at the
    # start position.
    defining_set = set()
    i = start
```

```
31
32      # add 2-cyclotomic cosets mod n to the defining set until the
33      # required minimal distance is reached or there are no
34      # further 2-cyclotomic cosets mod n.
35      while i <= n - 1:
36        defining_set = defining_set.union(cosets[i])
37        i += 1
38        achieved_bose_distance =
          ↪  _number_of_consecutive_elements(defining_set)
39
40        # the required minimal distance is achieved if there are
          ↪  more
41        # than required_min_distance consecutive elements in the
42        # defining set.
43        if achieved_bose_distance >= required_min_distance:
44
45          # the dimension of the code is n minus the size of the
46          # defining set (see [HP03, Theorem 4.4.2])
47          dimension = n - len(defining_set)
48          if dimension > min_dimension:
49            num_puncturing =
              ↪  _get_puncturing_amount(achieved_bose_distance,
              ↪  required_relative_minimal_bose_distance, n)
50
51            logger.debug("[{:d}, {:d}, {:d}]_2 at start {:d} and for
              ↪  {:d} cosets".format(n, dimension,
              ↪  achieved_bose_distance, start, i - start))
52            if n - num_puncturing < smallest_n:
53              smallest_n = n - num_puncturing
54              best_puncturing = num_puncturing
55              best_k = dimension
56              best_d = achieved_bose_distance
57
58              print("[{:d}, {:d}, {:d}] => {:d} x puncturing => [{:d},
                ↪  {:d}, {:d}]".format(n, dimension,
                ↪  achieved_bose_distance, num_puncturing, n -
                ↪  num_puncturing, dimension, achieved_bose_distance -
                ↪  num_puncturing))
59
60    print("Best: [{:d}, {:d}, {:d}] => {:d} x puncturing => [{:d},
      ↪  {:d}, {:d}]".format(n, best_k, best_d, best_puncturing, n -
      ↪  best_puncturing, best_k, best_d - best_puncturing))
61
62
```

```
63 def q_cyclotomic_cosets_mod_n(q, n):
64   """ Computes all q-cyclotomic cosets modulo n. The result is a
     ↪  dictionary mapping all 0 <= i <= n to the cyclotomic coset
     ↪  containing i."""
65   cosets = dict()
66   zero_coset = set()
67   zero_coset.add(0)
68   cosets[0] = zero_coset
69
70   for i in range(1, n):
71     if not i in cosets:
72       coset = set()
73       coset.add(i)
74       cosets[i] = coset
75       next_element = i * q % n
76       while not next_element == i:
77         coset.add(next_element)
78         cosets[next_element] = coset
79         next_element = next_element * q % n
80   return cosets
81
82
83 def _number_of_consecutive_elements(a):
84   """ Given the set a, the function outputs the largest number of
     ↪  consecutive elements in a"""
85
86   # A set of size zero always has zero consectutive numbers in it.
87   if len(a) == 0:
88     return 0
89
90   sorted_elements = sorted(list(a))
91
92   # the length of the longest sequence found so far
93   longest_sequence_length = 1
94   # the index where the longest sequence found so far starts (in
     ↪  the sorted list)
95   longest_sequence_start = 0
96   # the length of the currently considered sequence
97   current_sequence_length = 1
98   # the index where the currently considered sequence starts
99   current_sequence_start = 0
100
101  # initialize the previous element with the first element in order
     ↪  to emulate a do-while loop
```

```
102   previous_element = sorted_elements[0]
103   for i in range(1, len(sorted_elements)):
104     current_element = sorted_elements[i]
105     if current_element == previous_element + 1:
106       # if the current sequence is continued, update the length of
          ↪ the current
107       # sequence and, if it is longer than the currently longest
          ↪ sequence, update the information
108       # on the longest sequence found so far to the current
          ↪ sequence.
109       current_sequence_length += 1
110       if current_sequence_length > longest_sequence_length:
111         longest_sequence_length = current_sequence_length
112         longest_sequence_start = current_sequence_start
113       else:
114         # if the current element is not the continuation of
115         # the previous sequence, start a new sequence of length 1
116         current_sequence_start = i
117         current_sequence_length = 1
118       previous_element = current_element
119   return longest_sequence_length
120
121
122 def _get_puncturing_amount(achieved_bose_distance,
    ↪ required_rel_min_distance, code_out_length):
123   """ Given the achieved bose distance, the output length of an
      ↪ error correcting code and a desired relative minimal
      ↪ distance, this function computes how often the code can be
      ↪ punctured while still maintaining the desired relative
      ↪ minimal distance."""
124   return m.floor((achieved_bose_distance - code_out_length *
      ↪ required_rel_min_distance) / (1 - required_rel_min_distance))
```