The Dissertation Committee for Satyanarayana Vusirikala
certifies that this is the approved version of the following dissertation:

Constructing and Leveraging One-Way Function with Encryption

Committee:

Brent Waters, Supervisor

Hovav Shacham

Greg Plaxton

David Wu

Constructing and Leveraging One-Way Function with Encryption

by

Satyanarayana Vusirikala

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2021

Dedicated to my parents, Raju and Lakshmi.

Acknowledgments

My pursuit towards a doctoral degree has been an incredible journey that leads to significant intellectual enrichment and personal and professional development. Here I wish to thank the multitudes of people who helped me reach this wonderful milestone. I sincerely apologize to anyone whom I unintentionally forget to acknowledge.

First and foremost, I am extremely grateful to my esteemed supervisor, Prof. Brent Waters, for his invaluable advice, continuous support, and patience during my Ph.D. study. His immense knowledge and plentiful experience have encouraged me in my academic research and daily life. I would also like to thank Prof. Susan Hohenberger for her invaluable supervision, support, and tutelage during the course of my Ph.D. degree. I learned a lot from the incredible discussions on cryptographic automation I had with her.

I am also thankful to Rishab Goyal, Jiahui Liu for their numerous discussions regarding the problems related to the concepts of traitor tracing, lockable obfuscation, registration-based encryption, one-way function with encryption, multi-party computation, etc., and especially Rishab for his fruitful collaborations that led to the results obtained as part of this dissertation. I would like to especially thank Rishab Goyal for teaching me many things and being a wonderful friend and colleague. Without his tremendous understand-

ing, encouragement, and support in the past few years, it would be impossible for me to complete my study. He has been a major player in my fun-filled grad school life and was always available, leading to many enlightening conversations.

I also would like to thank all my past advisors and research mentors (Nishanth Chandran, Satya Lokam, Divya Gupta, Arnab Roy, Hart Montgomery, Saswata Shannigrahi, Payman Mohassel, Juan Garay) as well as the CS faculties at UT and IIT Guwahati for sharing their vast wealth of knowledge. I am grateful to Hovav Shacham, Greg Plaxton, and David Wu for being a part of my Ph.D. and RPE committees. I would like to particularly thank Nishanth Chandran, Satya Lokam, Susan Hohenberger, Arnab Roy, and Hart Montgomery for their inspiring discussions on technical topics and all the fellow students and colleagues across the globe whom I had the pleasure to interact with over the course of my doctoral studies (Rishab, Venkata, Ashutosh, Marilyn, Shashank, Swadhin, Andrew, Rachit, George, Sikhar, Gautam, and Jiahui). I would also like to express my gratitude to my roommates Gautam and Anand, who made my grad school experience filled with fun and joy.

A special thanks to Melinda (Lindy) Aleshire for being a wonderful person to interact with and taking care of all the administrative requirements. I am also very thankful to all the funding agencies (especially UT Austin provost fellowship) that have supported my research over the last several years and all the administrative staff at UT and all other places I have visited for research-related activities.

Finally, I wish to acknowledge everyone in my family for their consistent and unconditional support, love, and their insurmountable confidence and belief in my capabilities and a constant reminder of the most valuable things in life. I would like to therefore dedicate this dissertation to them, without whom none of this would have been possible.

Constructing and Leveraging One-Way Function with Encryption

Publication No. _____

Satyanarayana Vusirikala, Ph.D.
The University of Texas at Austin, 2021

Supervisor: Brent Waters

Over the last few years, there has been a surge of new cryptographic results, including Laconic oblivious transfer [30], (anonymous/ hierarchical) Identity-based Encryption [38, 42, 21], Trapdoor functions [46, 44, 39], Chosen-ciphertext security transformations [75, 74], Registration-Based Encryption [47, 48, 58], due to a beautiful framework recently introduced in the works of Cho et al. [30], and Döttling and Garg [38]. The primitive one-way function with encryption (OWFE) [46, 44] and its relatives (chameleon encryption, one-time signatures with encryption, hinting PRGs, trapdoor hash encryption, batch encryption) [38, 42, 21, 75, 41] have been a centerpiece in all these results.

While there exist multiple realizations of OWFE (and its relatives) from a variety of assumptions such as CDH, Factoring, and LWE, all such constructions fall under the same general "missing block" framework [30, 38]. Although this framework has been instrumental in opening up a new pathway towards

various cryptographic functionalities via the abstraction of OWFE (and its relatives), it has been accompanied by undesirable inefficiencies that might inhibit a much wider adoption in many practical scenarios. Motivated by the surging importance of the OWFE abstraction (and its relatives), a natural question to ask is whether the existing approaches can be diversified to obtain more constructions from different assumptions and develop newer frameworks. We believe answering this question will eventually lead to important and previously unexplored performance trade-offs in this novel cryptographic paradigm's overarching applications.

In this thesis, we propose a new *accumulation-style* framework for building a new class of OWFE constructions with a special focus on achieving shorter ciphertext size. Such performance improvements parlay into shorter parameters in their corresponding applications. Our OWFE constructions outperform in terms of ciphertext size and encryption time, but this comes at the cost of larger evaluation and setup times. We also provide concrete performance measurements for our constructions and compare them with existing approaches. We believe highlighting such trade-offs will lead to wider adoption of these abstractions in a practical sense.

In the second part of the thesis, we study an application of One-Way Function with Encryption. In Identity-based systems, there is a trusted authority that stores some secret information that gives him the ability to break the system's security if he turns malicious. To solve this problem, [47, 48] proposed the notion of Registration-Based Encryption (RBE) and construct it

from One-Way Function with Encryption and Garbling schemes. Here, each user of the system generates a public and secret key on their own and gives only their public information to the trusted authority. The trusted authority simply accumulates the public information collected from all the users in the system and publishes and master public key. The trusted authority does not store any secret information. However, a malicious authority could still harm the system by generating the master public key arbitrarily. We propose the notion of verifiable RBE, where any user of the system can obtain short and easily verifiable proof that the authority is doing its job honestly. We build verifiable RBE from One-Way Function with Encryption and Garbling. Our construction is more efficient than the earlier known constructions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 A Brief History of Cryptography

Cryptography is the art of hiding secrets. This art predates the conception of computers by many centuries. It can be traced to as early as ancient empires such as the Roman Empire. Many kings and emperors used cryptography to communicate with each other secretly. Many soldiers used cryptography to communicate with each other during wars. Traditionally, cryptographers hide secrets using the process of encryption, which recasts the sensitive information (called plaintext) into an unintelligible form called ciphertext. Cryptographers use the process of decryption to recover the plaintext from the ciphertext. Decades of research resulted in very efficient encryption and decryption procedures.

For many years, cryptographers believed that they must share some common secret information called a secret key for two parties to communicate covertly. This belief was dispelled by Diffie and Hellman [34] in 1976, who put forth a breakthrough concept called public-key cryptography. Here, two parties can secretly communicate without sharing any a priori agreed-upon mutual secret. In the public-key encryption, the receiver generates a pair of

public and secret keys. The receiver sends the public key to the sender. The sender then encrypts the message using the public key to generate a ciphertext. The receiver can decrypt the ciphertext using its secret key.

Goldwasser and Micali [56] formally defined what it is meant for a public-key encryption scheme to be secure. Since then, public-key encryption (PKE) has remained a cornerstone in modern-day cryptography. It has been one of the most widely used and studied cryptographic primitives. We use it every day to connect to a website or update software. When we access google.com, the google server sends its public key to our computer. Our computer then exchanges another secret key with google using this public key.

Over the last several decades, the cryptographic community has made significant research efforts in re-envisioning the original goals of hiding secrets and public-key encryption, pushing towards more expressiveness from such systems, thereby enabling newer applications. This lead to a new wave of cryptographic results. Yao [102, 101] proposed the notion of multi-party computation. Diffie and Hellman [34] proposed digital signatures. Rabin [87] proposed the notion of the oblivious transfer. Goldwasser et al. [57] conceived the notion of zero-knowledge proofs.

1.2 Identity-based Encryption

Shamir [94] introduced a novel type of cryptographic scheme, which enables any pair of users to communicate securely without exchanging private or public keys. He called the system Identity-based Encryption (IBE). It is

a type of public-key encryption in which the public key of a user is some unique information about the user's identity, such as the user's email address. This means that a sender who can access the system's public parameters can encrypt a message using the receiver's identity, such as the email address as a key. The receiver obtains its decryption key from a central authority, which needs to be trusted as the central authority generates secret keys for every user.

Boneh and Franklin [15] gave the first construction of Identity-based Encryption (IBE). In cryptography, it is hard to prove the security of a scheme unconditionally. If we can prove that a cryptographic construction is unbreakable by any polynomial-time attacker, it means we found an algorithm which is in NP but not in P, thereby proving that $\mathsf{P} \neq \mathsf{NP}$. Consequently, we build various cryptographic schemes assuming there are hard problems that cannot be solved by any attacker in polynomial time. Some of the hard problems include factoring, quadratic residuosity, discrete log, decisional Diffie-Hellman, learning with errors and many other pairing-based hard problems. Many researchers tried to solve these problems in the past but had only partial progress. As a result, these problems are widely believed to be hard, although there is no formal proof.

Boneh and Franklin [15] based on their construction of IBE on mathematical structure called pairings. In this construction, they assumed the existence of three multiplicative cryptographic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, along with a pairing map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with some special properties. Later, many

constructions with various security trade-offs emerged. These constructions were based on wide variety of assumptions such as pairing-based assumptions [25, 13, 98, 99, 77], quadratic residuosity [32] and lattices [53, 26, 1].

1.3 The Rise of One-Way Function with Encryption

Constructing IBE from cryptographic groups without the above pairing map has long been an open problem. Recently, Döttling and Garg [38] solved this long-standing open problem by building IBE from pairing-free groups based on Computational Diffie-Hellman (CDH) assumption. Döttling and Garg [38] built upon techniques proposed by Cho et al. [30] and abstracted out the technique into a cryptographic primitive called Chameleon Encryption. They showed how to construct chameleon encryption based on pairing-free groups (using computational Diffie-Hellman assumption). They also showed how to use chameleon encryption to build Identity-based Encryption.

Many works [50, 42, 21, 51, 10, 46, 47, 39, 44, 75, 48] later on showed that chameleon encryption is a very useful primitive in cryptography. These works proposed variants of chameleon encryption which are slightly different in syntax and security properties. Some variants include one-way function with encryption (OWFE), hash encryption, one-time signatures with encryption, hinting PRGs, trapdoor hash encryption, and batch encryption. These works also used these variants to construct advanced cryptographic functionalities. We list some of these applications here.

Garg and Srinivasan[50, 51] used the techniques to build a 2-round

multi-party computation protocol. Benhamouda and Lin[10] used the techniques to construct a k-round multi-party computation protocol based on k-Round oblivious transfer. [42, 21] constructed IBE from slightly weaker primitives called One-Time Signatures with Encryption and Batch Encryption, respectively. [46, 44, 39] constructed Trapdoor Functions and Deterministic Encryption based on a related primitive called One-Way Function with Encryption. [75] showed how to amplify CPA security of a PKE/ABE scheme to CCA security using a related primitive called Hinting PRGs. [47, 48] introduced the notion of Registration-based encryption as a means to solve the key-escrow problem in Identity-based systems. They built this novel encryption scheme using Batch Encryption and garbling. [79, 86, 72] used the signaling techniques in [75] to build Designated-verifier zero-knowledge proofs. [41] constructed 2-message rate-1 string oblivious transfer based on Trapdoor Hash Functions.

Most of these works have been propelled by the primitive of one-way function with encryption (OWFE) [46, 44] and its relatives (Chameleon Encryption, Hash Encryption, One-time signatures with Encryption, Hinting PRGs, Trapdoor Hash Encryption, Batch Encryption) [38, 42, 21, 75, 41]. One-Way Function with Encryption is known to imply many primitives such as Batch Encryption, Hash Encryption, One-Time Signatures with Encryption [40]. This thesis also shows that it implies Hinting PRGs [59]. Suppose the one-way function with encryption scheme has small parameters, a.k.a the succinctness criteria. In that case, the resultant Batch Encryption, Hash En-

cryption, One-Time Signatures with Encryption also have small parameters, i.e., satisfies succinctness.

As a succinct one-way function with encryption forms a centerpiece of a great line of work, there is a clear need to find more efficient constructions of the primitive. Such a result would improve the efficiency of many of the above results. For example, it would improve the efficiency of trapdoor functions, identity-based encryption, registration-based encryption, and CPA to CCA transformations. This thesis proposes new constructions of succinct one-way function with encryption. In the rest of the section, we provide a formal definition of the one-way function with encryption and describe limitations in the existing frameworks used for constructing the primitive.

1.4 One-way Function with Encryption Definition

Let us first review the notion of the family of one-way functions. The family of functions $\mathcal{F} = \{f_{\mathsf{pp}}\}_{\mathsf{pp}}$ is parameterized by public parameters pp which are sampled during setup. The function is said to be one-way if it can be computed in polynomial time and cannot be inverted by any polynomial-time algorithm. A one-way function with encryption scheme extends the fundamental notion of one-way functions in the following way. The scheme is associated with a family of one-way functions $\mathcal{F} = \{f_{\mathsf{pp}}\}_{\mathsf{pp}}$ where during setup one samples public parameters pp that fixes the underlying one-way function $f = f_{\mathsf{pp}}$.

The special feature of an OWFE scheme, which makes them such a

useful primitive, is the possibility of performing encryption and decryption without sampling additional keys. Formally, in an OWFE scheme, the encryption procedure is abstracted into two components — algorithms E_1, E_2 which work as follows. Both E_1 and E_2 share the same random coins ρ, and take as inputs a value y (that lies in the image space of f), an index-bit pair (i, b), and parameters pp. Algorithm E_1 is used to compute the "ciphertext" ct, whereas E_2 computes a random KEM key k [1]. The decryption algorithm D on input a ciphertext ct, pre-image string x, and parameters pp, outputs a decrypted key k'.

The scheme should satisfy certain correctness and security requirements. For correctness, it is important that if the string x is such that $y = f_{\mathsf{pp}}(x)$ and i^{th} bit of x equals b, then D should decrypt correctly i.e., $k' = k$. While for security, we require the unpredictability of the one-way function f and that the ciphertext does not leak the key k trivially. That is, given an input x, parameters pp and a ciphertext ct, the associated key k must be indistinguishable from random as long as we perform the encryption for some value $y = f_{\mathsf{pp}}(x)$ and any index-bit pair of the form $(i, 1-x_i)$. Various other properties have been studied for OWFE schemes, such as recyclability and smoothness since they lead to more applications. For this introduction, we

[1] In cryptography, it is quite common for an encryption algorithm to compute a random key k. We can then use the key k to encrypt the given message m using a regular secret key encryption. The decryption algorithm also computes the key k given the ciphertext. We then decrypt the secret key encryption of message m using the key k. Such a mechanism is called the Key Encapsulation Mechanism (KEM). We refer to the key k as a KEM key throughout this thesis.

only focus on the above simpler properties. We provide further details later. [2]

Intuitively, an OWFE scheme is said to be succinct if the length of OWF value $f(x)$ is a polynomial in security parameter and is independent of the size of the input x. A OWFE scheme is simply a one-way function f equipped with matching encryption-decryption procedures such that encryption allows encrypting random keys with respect to an OWF output string y and a pre-image bit (i, b), while decryption requires a pre-image x such that $f(x) = y$ and $x_i = b$.

1.5 The "Missing Block" Framework

While there exist multiple realizations of OWFE (and its relatives) from a variety of assumptions such as CDH, Factoring, and LWE, all such constructions fall under the same general "missing block" framework [30, 38]. To illustrate the aforementioned framework, we sketch the OWFE construction provided by Garg and Hajiabadi [46]. The construction is based on a cryptographic group $\mathbb{G}$ in which the computational Diffie-Hellman problem is hard to solve. In other words, we assume that given a random group generator g and 2 group elements g^c, g^d, where c, d are sampled randomly, no probabilistic polynomial-time algorithm can compute $g^{c \cdot d}$ with non-negligible probability.

Let the input length of the one-way function be n. The public parame-

[2] Briefly, the recyclability property says that the E_1 algorithm does not depend on the value y. The smoothness property says that given the output $y = f(x)$ of the OWF, the corresponding pre-image x is unpredictable as long as we draw x from a distribution with sufficient min-entropy.

ters consists of $2n$ randomly sampled group generators $\{g_{i,b}\}_{(i,b)\in[n]\times\{0,1\}}$. The function output is computed by performing subset-product on the public parameters, where the subset selection is done as per the input bits. Concretely, on an input $x \in \{0,1\}^n$, the output is $f(x) = \prod_i g_{i,x_i}$. The ciphertext structurally looks like the public parameters, that is it also consists of $2n$ group elements $\{c_{i,b}\}_{i,b}$. Here to encrypt to pre-image bit (i^*, b^*) under randomness ρ, the encryption algorithm E_1 simply sets $c_{i,b} = g_{i,b}^{\rho}$ for all $(i,b) \neq (i^*, 1-b^*)$, with the $(i^*, 1-b^*)^{th}$ term not being set (i.e., $c_{i^*,1-b^*} = \perp$). Pictorially, this can be represented as follows (where $i^* = 2$ and $b^* = 0$):

$\mathsf{pp} =$

$g_{1,0}$	$g_{2,0}$	$g_{3,0}$	$\cdots$	$g_{n,0}$
$g_{1,1}$	$g_{2,1}$	$g_{3,1}$	$\cdots$	$g_{n,1}$

$\xrightarrow[E_1(\mathsf{pp},(2,0);\rho)]{\text{Encryption}}$ $\mathsf{ct} =$

$g_{1,0}^{\rho}$	$g_{2,0}^{\rho}$	$g_{3,0}^{\rho}$	$\cdots$	$g_{n,0}^{\rho}$
$g_{1,1}^{\rho}$	$\times$	$g_{3,1}^{\rho}$	$\cdots$	$g_{n,1}^{\rho}$

The encryptor computes the KEM key as $\mathsf{HC}(y^{\rho})$, where y is the output of the OWF, and HC corresponds to the hardcore predicate. On the other hand, the decryptor does not know the randomness ρ. Thus given the ciphertext ct and a valid pre-image x, it computes the subset-product on ct (followed by applying the hardcore predicate), where it selects the subset based on x. That is, the decryptor computes the key as $\mathsf{HC}(\prod_i c_{i,x_i})$.

This notion of not setting up the $(i^*, 1-b^*)^{th}$ term in the ciphertext is what we refer to as adding a "missing block". The intuition behind this is that the ciphertext should only be decryptable using pre-images x such that $x_{i^*} = b^*$, thus the ciphertext component corresponding to the pre-image bit $(i^*, 1-b^*)$ can be omitted. Here the omission of the $(i^*, 1-b^*)^{th}$ block is crucial in proving the security of encryption.

1.5.1 Limitations of the Framework

The "missing block" framework has been instrumental in opening up a new pathway towards various cryptographic functionalities via the abstraction of OWFE (and its relatives). However, the framework resulted in undesirable inefficiencies that have led to large system parameters in most of the applications. In particular, the OWFE described above in this framework leads to large "ciphertexts" where the size grows linearly with the input length n of the OWF. Now, this inefficiency gets amplified differently in each of its applications. For instance, large OWFE ciphertexts lead to considerably large public parameters of a trapdoor function (deterministic encryption) [46, 44]. This is because the public parameters as per those transformations consist of a polynomial number of OWFE ciphertexts that grow linearly with n. Similar situations arise if we look at the Hinting PRG abstraction [75], where the existing constructions via the "missing block" framework leads to much worse public parameters. The performance overhead also gets significantly amplified if we look at its application to chosen-ciphertext security transformations [75].

1.6 Summary of our Results

In this section, we present the summary of our results. We postpone the further details to the main body of the paper. We also published all the results presented in this thesis at [59, 58].

Motivated by the surging importance of the abstraction of one-way function with encryption and its relatives, a natural question is whether the

existing approaches can be diversified to obtain more constructions from different assumptions and develop newer frameworks. We believe answering this question will eventually lead to significant and previously unexplored performance trade-offs in this novel cryptographic paradigm's overarching applications. In this thesis, we present an alternative framework to build a one-way function with encryption, which we call the "accumulator framework". Using this technique, we present 3 constructions of one-way function with encryption that are more efficient than the prior constructions based on some metrics (ciphertext size and encryption time). We present an overview of the performance evaluation in Table 1.1. We show that a one-way function with encryption can be used to construct hinting prg generically. Additionally, we provide 2 new constructions of hinting prg using our accumulator framework.

In this thesis, we also present an application of One-Way Function with Encryption called "verifiable registration-based encryption". In a typical identity-based encryption system, there is a central authority that generates secret keys for each individual user. The central authority has the ability to decrypt any ciphertext in the system. This leads to a single point of failure and poses a huge threat to the system. To defend against a malicious central authority, Garg et al. [47] proposed a novel encryption scheme called "registration-based encryption". This system shares identity-based properties. That is, the encryptor can encrypt a message using the receiver's identity. At the same time, the central authority does not pose a threat. They built this system using "hash garbling, " which can be constructed from a one-way func-

tion with encryption. In this thesis, we argue that registration-based encryption does not solve the issue with central authority entirely. We thereby add a property called "verifiability" and define verifiable registration-based encryption. We present a construction of this primitive based on hash garbling. We discuss more details in Chapter 7.

Metric	DDH [46]	Φ-Hiding (§5.1)	DDHI (§5.2)	DBDHI (§5.3)
pp Size	65.4 KB	321.8 KB	32.8 KB	60.8 KB
ct Size	65.4 KB	384 Bytes	32.8 KB	192 Bytes
Key Size	64 Bytes	384 Bytes	64 Bytes	1146 Bytes
Time (Setup)	0.016s	12.43s	0.070s	0.158s
Time (f)	0.0002s	3.67s	0.090s	0.19s
Time (E_1)	134.90ms	9.40ms	76.40ms	0.45ms
Time (E_2)	0.14ms	8.38ms	0.136ms	1.79ms
Time (D)	0.0003s	3.57s	0.090s	0.19s

Table 1.1: Concrete performance evaluation of various OWFE constructions for 128-bit security level

1.7 Organization

We now provide a roadmap of the thesis. In Chapter 2, we define various primitives and assumptions that will be helpful throughout the thesis. In Chapter 3, we provide a detailed overview of our techniques related to the constructions of one-way function with encryption and hinting prgs. In Chapter 4, we build the mathematical background and theorems required for our constructions. In Chapter 5, we finally describe our one-way function with encryption and hinting prg constructions. In Chapter 6, we describe the implementation details of our constructions and compare the performance with previous works.

In Chapter 7, we provide a detailed overview of an application of a one-way function with encryption called verifiable registration-based encryption. In Chapter 8, we provide a formal definition of verifiable registration based encryption. In Chapter 9, we describe our construction of verifiable registration-based encryption and prove its security.

Chapter 2

Background

In this chapter, we first discuss various notations used in this thesis. We then review the definitions of one-way function with encryption, hinting prgs, and strong extractors. We then describe various cryptographic assumptions we come across in this thesis.

Notations. Let PPT denote probabilistic polynomial-time. We denote the set of all positive integers up to n as $[n] := \{1, \ldots, n\}$. Unless specified, all polynomials we consider are positive polynomials throughout this paper. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. We use the distribution $\mathcal{D}^n$ to represent a distribution over vectors of n components. Here, each component is drawn independently from the distribution $\mathcal{D}$. We call any distribution on n-length bit strings with minimum entropy k as a (k, n) source.

2.1 One-Way Function with Encryption

Here we recall the definition of a recyclable one-way function with encryption from [46, 44]. We adapt the definition to a setting where the KEM key is an ℓ-bit string instead of just a single bit. A recyclable (k, n, ℓ)-OWFE scheme consists of the PPT algorithms K, f, E_1, E_2 and D with the following syntax.

$K(1^\lambda) \to \mathsf{pp}$: Takes the security parameter 1^λ and outputs public parameters pp.

$f(\mathsf{pp}, x) \to y$: Takes a public parameter pp and a preimage $x \in \{0,1\}^n$, and deterministically outputs y.

$E_1(\mathsf{pp}, (i, b); \rho) \to \mathsf{ct}$: Takes public parameters pp, an index $i \in [n]$, a bit $b \in \{0,1\}$ and randomness ρ, and outputs a ciphertext ct.

$E_2(\mathsf{pp}, y, (i, b); \rho) \to k$: Takes a public parameter pp, a value y, an index $i \in [n]$, a bit $b \in \{0,1\}$ and randomness $\rho \in \{0,1\}^r$, and outputs a key $k \in \{0,1\}^\ell$. Notice that unlike E_1, which does not take y as input, the algorithm E_2 does take y as input.

$D(\mathsf{pp}, \mathsf{ct}, x) \to k$: Takes a public parameter pp, a ciphertext ct, a preimage $x \in \{0,1\}^n$, and deterministically outputs a key $k \in \{0,1\}^\ell$.

We require the following properties.

Correctness. For security parameter λ, for any choice of $\mathsf{pp} \in K(1^\lambda)$, any index $i \in [n]$, any preimage $x \in \{0,1\}^n$ and any randomness value ρ, the following holds: letting $y := f(\mathsf{pp}, x)$, and $\mathsf{ct} := E_1(\mathsf{pp}, (i, x_i); \rho)$, we have $E_2(\mathsf{pp}, y, (i, x_i); \rho) = D(\mathsf{pp}, \mathsf{ct}, x)$.

Definition 2.1.1 ((k,n)-One-wayness)**.** *For any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have*

$$\Pr\left[f(\mathsf{pp}, \mathcal{A}(\mathsf{pp}, y)) = y \ : \ \begin{array}{c} S \leftarrow \mathcal{A}(1^\lambda); \mathsf{pp} \rightarrow K(1^\lambda); \\ x \leftarrow S; y = f(\mathsf{pp}, x) \end{array}\right] \leq negl(\lambda).$$

Here, the adversary is constrained to output only a (k,n)-source.

Definition 2.1.2 (Security for encryption)**.** *For any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have*

$$\Pr\left[\begin{array}{l} \\ \mathcal{A}(\mathsf{pp}, x, \mathsf{ct}, k_b) = b \ : \end{array} \begin{array}{c} (x,i) \leftarrow \mathcal{A}(1^\lambda); \mathsf{pp} \leftarrow K(1^\lambda); \\ b \leftarrow \{0,1\}; \rho \leftarrow \{0,1\}^r; \\ \mathsf{ct} \leftarrow E_1(\mathsf{pp}, (i, 1-x_i); \rho); \\ k_0 \leftarrow E_2(\mathsf{pp}, f(\mathsf{pp}, x), (i, 1-x_i); \rho); \\ k_1 \leftarrow \{0,1\}^\ell \end{array}\right] \leq 1/2 + negl(\lambda).$$

Definition 2.1.3 ((k,n)-Smoothness)**.** *We say that (K, f, E_1, E_2, D) is (k,n)-smooth if for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$, such that for all $\lambda \in \mathbb{N}$, we have*

$$\Pr\left[\mathcal{A}(\mathsf{pp}, y) = b \ : \ \begin{array}{c} (S_0, S_1) \leftarrow \mathcal{A}(1^\lambda); \mathsf{pp} \leftarrow K(1^\lambda); \\ b \leftarrow \{0,1\}; x_0 \leftarrow S_0; x_1 \leftarrow S_1; y = f(\mathsf{pp}, x_b) \end{array}\right] \leq 1/2 + negl(\lambda).$$

where the distributions S_0 and S_1 output by the adversary $\mathcal{A}$ are constrained to be (k,n)-sources.

2.2 Hinting PRG

In this section, we review the definition of Hinting PRG proposed in [75]. Let $n(\cdot)$ and $\ell(\cdot)$ be some polynomials. An (n, ℓ)-hinting PRG scheme consists of two PPT algorithms Setup, Eval with the following syntax.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, n)$: The setup algorithm takes as input the security parameter λ, and length parameter ℓ, and outputs public parameters pp and input length $n = n(\lambda)$.

$\mathsf{Eval}(\mathsf{pp}, s \in \{0,1\}^n, i \in [n] \cup \{0\}) \to y \in \{0,1\}^\ell$: The evaluation algorithm takes as input the public parameters pp, an n-bit string s, an index $i \in [n] \cup \{0\}$ and outputs an ℓ bit string y.

Definition 2.2.1. *An (n, ℓ)-hinting PRG scheme $(\mathsf{Setup}, \mathsf{Eval})$ is said to be secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$\Pr\left[\mathcal{A}\Big(\mathsf{pp}, y_0^\beta, \{y_{i,b}^\beta\}_{i\in[n],b\in\{0,1\}}\Big) = \beta \;:\; \begin{array}{c} (\mathsf{pp}, n) \leftarrow \mathsf{Setup}(1^\lambda); \\ s \leftarrow \{0,1\}^n; \beta \leftarrow \{0,1\}; \\ y_0^0 = \mathsf{Eval}(\mathsf{pp}, s, 0); y_0^1 \leftarrow \{0,1\}^\ell; \\ y_{i,s_i}^0 = \mathsf{Eval}(\mathsf{pp}, s, i); \\ y_{i,\overline{s_i}}^0 \leftarrow \{0,1\}^\ell \; \forall i \in [n]; \\ y_{i,b}^1 \leftarrow \{0,1\}^\ell \; \forall i \in [n], b \in \{0,1\} \end{array}\right] \leq 1/2 + negl(\lambda)$$

2.3 Strong Extractors

Extractors are combinatorial objects used to 'extract' uniformly random bits from a high entropy source. In this thesis, we will be using seeded

extractors. In a seeded extractor, the extraction algorithm takes as input a sample point x from the high randomness source $\mathcal{X}$, together with a short seed $\mathfrak{s}$, and outputs a string that looks uniformly random. Here, we will be using strong extractors, where the extracted string looks uniformly random even when the seed is given.

Definition 2.3.1. *A (k, ϵ) strong extractor $\mathsf{Ext} : \mathbb{D} \times \mathbb{S} \to \mathbb{Y}$ is a deterministic algorithm with domain $\mathbb{D}$, range $\mathbb{Y}$ and seed space $\mathbb{S}$ such that for every source $\mathcal{X}$ on $\mathbb{D}$ with min-entropy at least k, the following two distributions have statistical distance at most ϵ:*

$$\mathcal{D}_1 = \{(\mathfrak{s}, \mathsf{Ext}(x, \mathfrak{s})) : \mathfrak{s} \leftarrow \mathbb{S}, x \leftarrow \mathcal{X}\}, \mathcal{D}_2 = \{(\mathfrak{s}, y) : \mathfrak{s} \leftarrow \mathbb{S}, y \leftarrow \mathbb{Y}\}$$

Using the Leftover Hash Lemma, we can construct strong extractors from pairwise-independent hash functions. More formally, let $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of pairwise independent hash functions, and let $m = k - 2\log(1/\epsilon)$. Then $\mathsf{Ext}(x, h) = h(x)$ is a strong extractor with h being the seed. Such hash functions can be represented using $O(n)$ bits.

2.4 Computational Assumptions

In this section, we review various cryptographic assumptions we come across in this thesis.

2.4.1 Φ-Hiding Assumption

The Φ-Hiding assumption, introduced by Cachin et al. [22], informally states that given an RSA modulus N, it is hard to find the factors of $\phi(N)$, or to distinguish a factor of $\phi(N)$ from an integer co-prime to $\phi(N)$. To formally state this assumption, we need to introduce some notations, and will be following the work of [67] for the same. Let $\text{PRIMES}(\lambda)$ denote the set of primes of bit-length λ, and let

$$\mathsf{RSA}(\lambda) = \{N : N = pq;\ p, q \in \text{PRIMES}(\lambda/2);\ gcd(p-1, q-1) = 2\}.$$

For any $e \leq 2^\lambda$, let

$$\mathsf{RSA}_e(\lambda) = \{N \in \mathsf{RSA}(\lambda) : e \text{ divides } \phi(N)\}.$$

Assumption 1 (Φ-Hiding)**.** *The Φ-Hiding assumption states that for all $\epsilon > 0$, integers e such that $3 < e < 2^{\lambda/4-\epsilon}$ and PPT adversaries $\mathcal{A}$,*

$$\Pr[\mathcal{A}(N, e) = 1 : N \leftarrow \mathsf{RSA}(\lambda)] - \Pr[\mathcal{A}(N, e) = 1 : N \leftarrow \mathsf{RSA}_e(\lambda)] \leq negl(\lambda).$$

2.4.2 CDH Assumption

Computational Diffie-Hellman (CDH) [34] assumption is one of the most well-studied assumptions in cryptography. We say that a PPT algorithm GGen is a group generator if it takes a security parameter 1^λ as input and outputs a "group description" $\mathcal{G} := (\mathbb{G}, p)$. Here $\mathbb{G}$ is a group with prime order $p = \Omega(2^\lambda)$, from which one can efficiently sample a generator uniformly at random.

Assumption 2 (CDH)**.** *Let* GGen *be a group generator and* $q = q(\lambda) = \mathsf{poly}(\lambda)$*. We say that CDH assumption holds with respect to* GGen *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, we have*

$$\Pr\left[\mathcal{A}(\mathcal{G}, g, g^{\alpha}, g^{\beta}) = g^{\alpha\cdot\beta} \ : \ \begin{array}{c}(\mathbb{G}, p) = \mathcal{G} \leftarrow \mathsf{GGen}(1^{\lambda});\\ g \leftarrow \mathbb{G}; \alpha, \beta \leftarrow \mathbb{Z}_p^*\end{array}\right] \leq 1/2 + negl(\lambda).$$

2.4.3 q-DDHI Assumption

q-DDHI is one of the well-studied assumptions. Boneh and Boyen[13] first introduced a variant of this assumption. We say that a PPT algorithm GGen is a group generator if it takes a security parameter 1^{λ} as input and outputs a "group description" $\mathcal{G} := (\mathbb{G}, p)$. Here $\mathbb{G}$ is a group with prime order $p = \Omega(2^{\lambda})$, from which one can efficiently sample a generator uniformly at random.

Assumption 3 (q-DDHI)**.** *Let* GGen *be a group generator and* $q = q(\lambda) = \mathsf{poly}(\lambda)$*. We say that q-Decisional Diffie Hellman Inversion assumption holds with respect to* GGen *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, we have*

$$\Pr\left[\mathcal{A}(\mathcal{G}, g, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^q}, T_b) = b \ : \ \begin{array}{c}(\mathbb{G}, p) = \mathcal{G} \leftarrow \mathsf{GGen}(1^{\lambda});\\ g \leftarrow \mathbb{G}; \alpha, r \leftarrow \mathbb{Z}_p^*\\ b \leftarrow \{0,1\}; T_0 = g^{1/\alpha};\\ T_1 \leftarrow g^r\end{array}\right] \leq 1/2 + negl(\lambda).$$

2.4.4 q-DBDHI Assumption

This assumption is introduced by Boneh and Boyen [13]. We say that a PPT algorithm GGen is a group generator if it takes a security parameter 1^λ as input and outputs a "group description" $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_T, e, p)$. Here, $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups with prime order $p = \Omega(2^\lambda)$, from which one can efficiently sample a generator uniformly at random. $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is an efficiently computable pairing operation.

Assumption 4 (q-DBDHI)**.** *Let* GGen *be a group generator and* $q = q(\lambda) = \mathsf{poly}(\lambda)$*. We say that q-Decisional Bilinear Diffie Hellman Inversion assumption holds with respect to* GGen *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, we have*

$$\Pr\left[\mathcal{A}(\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^q}, T_b) = b \ : \begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, e, p) = \mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda); \\ g \leftarrow \mathbb{G}_1; \alpha, r \leftarrow \mathbb{Z}_p^* \\ b \leftarrow \{0,1\}; T_0 = e(g,g)^{1/\alpha}; \\ T_1 \leftarrow e(g,g)^r\end{array}\right] \leq 1/2 + negl(\lambda).$$

Chapter 3

Overview of One-Way Function with Encryption

In this chapter, we provide a detailed overview of the techniques developed for building efficient one-way function with encryption and hinting prg systems. First, we describe the OWF with Encryption from Φ-hiding assumption and compare its efficiency with the prior work. We then describe the Hinting PRG construction based on the Φ-hiding assumption and compares its efficiency with the prior results. We then discuss the limitations of the constructions based on Φ-hiding assumptions. We finally provide the overview of the constructions based on q-DDHI and q-DBDHI assumptions. Finally, we conclude the chapter by discussing some related works.

In this thesis, we develop a new *accumulation-style* framework for building a new class of one-way function with encryption (as well as hinting PRG) constructions with a particular focus on achieving shorter ciphertext size (and shorter public parameter size, respectively). This improvement will parlay into shorter parameters in their corresponding applications. Concretely we explore the following performance trade-offs. For OWFE, our constructions based on this new framework outperform the existing ones in terms of ciphertext size

and encryption time, but this comes at the cost of higher evaluation and setup times. In terms of the applications of OWFE to deterministic encryption, this trade-off translates to a scheme with much smaller public parameters and setup time but larger encryption/decryption times. For hinting PRGs, our constructions provide a rather dramatic trade-off between evaluation time versus parameter size compared to prior schemes, with our construction leading to significantly shorter public parameter size. In terms of applications of hinting PRG to chosen-ciphertext security transformations [75], the trade-off between public parameter size and evaluation time in the hinting PRG constructions carries forward to a trade-off between encryption key/ciphertext sizes and encryption/decryption times in the resultant CCA-secure construction. Next, we describe our constructions' main ideas, and later we give some concrete performance metrics.

3.1 OWF with Encryption from Φ-Hiding Assumption

Our new framework's main inspiration are the number theoretic-based accumulators [9, 6, 92, 24, 54, 84, 23, 3, 27] that have been widely studied in the literature. For building an OWFE scheme from RSA-family assumptions, we look back at the RSA-based accumulators [9, 6, 92, 24, 54] which are the earliest number-theoretic instantiations available. Briefly, our idea is as follows. During setup, we sample n pairs of large primes $\{e_{i,b}\}_{i,b}$ and a random generator g in the base group of the RSA modulus ($\mathbb{Z}_N^*$). The one-way function associated with those parameters will correspond to an accumulation of half

of these primes, depending upon the input x, with respect to generator g. A ciphertext for a pre-image bit (i, b) will look like an accumulation of only the $(i, b)^{th}$ prime, i.e., $e_{i,b}$. During decryption, the decryptor accumulates the rest of the primes per the remaining pre-image bits. Below we sketch our construction in more detail.

The public parameters pp consist of an RSA modulus N, n pairs of λ-bit primes $\{e_{i,b}\}_{(i,b)\in[n]\times\{0,1\}}$, a generator $g \in \mathbb{Z}_N^*$, and a pairwise independent hash H. (Here n is the input length.) Given an input $x \in \{0,1\}^n$, the one-way function $f_{\mathsf{pp}}(x)$ is computed as $g^{H(x)\cdot\prod_i e_{i,x_i}}(\mod N)$. The encryption algorithm E_1 on input a pre-image bit (i^*, b^*) and randomness ρ, outputs ciphertext as $\mathsf{ct} = g^{\rho\cdot e_{i^*,b^*}}(\mod N)$.[1] The corresponding KEM key is set as $k = y^\rho(\mod N)$, where y is the output of the OWF. Lastly, the decryption procedure given a ciphertext ct and a pre-image x such that $f_{\mathsf{pp}}(x) = y$ and $x_{i^*} = b^*$, computes the key as $k' = \mathsf{ct}^{\prod_{i\neq i^*} e_{i,x_i}}(\mod N)$. Next, we briefly sketch the main arguments behind the security of this construction.

One-wayness Property. The one-wayness argument proceeds as follows. Suppose an adversary finds a collision $x \neq x'$, i.e. $f_{\mathsf{pp}}(x) = f_{\mathsf{pp}}(x')$, then a reduction algorithm can sample the λ-bit primes in such a way that, as long as n is larger than $\log N + \lambda$, it can break RSA assumption for one of the primes sampled as part of the public parameters. For proving security of encryption, we need to slightly modify the construction wherein we need to

[1] Technically, the ciphertext should also include the index i^* but we drop it for ease of exposition.

apply an extractor on the KEM key to prove it looks indistinguishable from random, that is $k = \mathsf{Ext}(\mathfrak{s}, y^{\rho})$ where Ext is a strong seeded extractor and seed $\mathfrak{s}$ is sampled during setup.

Security of Encryption. Recall that security of encryption requires that for any index-bit pair (i^*, b^*) and input x such that $x_{i^*} \neq b^*$, given a ciphertext $\mathsf{ct} = E_1(\mathsf{pp}, (i^*, b^*); \rho)$ the associated KEM key $k = E_2(\mathsf{pp}, f_{\mathsf{pp}}(x), (i^*, b^*); \rho)$ must be indistinguishable from random. The idea behind proving the same for the above construction is the following — ciphertext looks like $\mathsf{ct} = g^{\rho \cdot e_{i^*,b^*}}$ whereas the key is computed as $k = \mathsf{Ext}(\mathfrak{s}, g^{\rho \cdot \prod_i e_{i,x_i}})$. Since $b^* \neq x_{i^*}$, thus the key can be re-written as $k = \mathsf{Ext}(\mathfrak{s}, (\mathsf{ct}^{\prod_i e_{i,x_i}})^{e^{-1}_{i^*,b^*}})$. Now under the Φ-hiding assumption, we can argue that an adversary can not distinguish between the cases where e_{i^*,b^*} is co-prime with respect to $\phi(N)$, and when e_{i^*,b^*} divides $\phi(N)$. Note that in the latter case, there are e_{i^*,b^*} many distinct $e^{th}_{i^*,b^*}$ roots of $\mathsf{ct}^{\prod_i e_{i,x_i}}$. Thus, by strong extractor guarantee we can conclude that key k looks uniformly random to the adversary as the underlying source has large (λ bits of) min-entropy.

Smoothness Property. Lastly, we show that the above scheme satisfies the smoothness property for appropriate parameters ℓ, n. Formally, the (ℓ, n)-smoothness property says that for any two (ℓ, n)-sources S_0 and S_1, the distributions $\{f_{\mathsf{pp}}(x) : x \leftarrow S_0\}$ and $\{f_{\mathsf{pp}}(x) : x \leftarrow S_1\}$ should be computationally indistinguishable. A natural approach to proving smoothness is first to show that the function $t(x) = H(x) \cdot \prod_i e_{i,x_i} (\mathsf{mod}\ \phi(N))$ is a 2-universal hash function, and then apply Leftover Hash Lemma (LHL) to argue that

$t(x)$ is statistically close to uniform when we sample x from an appropriate source S. Note that if we could prove this, then smoothness property would follow directly since we know that the distributions $\{g^{t(x)} : x \leftarrow S_0\}$ and $\{g^{t(x)} : x \leftarrow S_1\}$ will be indistinguishable if $\{t(x) : x \leftarrow S_0\}$ and $\{t(x) : x \leftarrow S_1\}$ are indistinguishable.

Although prior works employed similar strategies, such an approach does not work in this case. This is because, in our construction, the exponents $\{e_{i,b}\}$ are sampled as random λ-bit primes instead of being sampled uniformly from $\mathbb{Z}_{\phi(N)}$. Thus we can not show $t(\cdot)$ to be a 2-universal hash function. We devise novel number-theoretic techniques to prove smoothness while getting around the above bottleneck in this thesis. Our approach is to provide a tight LHL-style proof at a very high level, which allows us to argue partial statistical indistinguishability. To complete the argument, we rely on the computational hardness of the Φ-hiding assumption. Our proof is a mixture of a computational and statistical argument. Below, we describe the structure at a high level.

Let $r_1 \cdot r_2 \cdots r_m$ (for some m) denote the prime factorization of $\phi(N)$, with $r_1 > r_2 > \cdots > r_m$.[2] Now suppose that a PPT adversary $\mathcal{A}$ can distinguish between distributions $\{g^{t(x)} : x \leftarrow S_0\}$ and $\{g^{t(x)} : x \leftarrow S_1\}$ with non-negligible probability ϵ. We show how to use such an adversary to break the Φ-hiding assumption with non-negligible probability.

[2]For ease of exposition, we assume that all the prime factors $\{r_i\}_i$ are distinct. In the main body, we do not make this restriction and present a more general proof.

The proof proceeds in two steps, wherein the first step we argue that $(t(x) \bmod r_i)$ is statistically close to random over $\mathbb{Z}_{r_i}$ for all prime factors r_i greater than a fixed threshold $\tau_{N,\epsilon}$. By a very tight LHL-style proof, we show that if we set the threshold $\tau_{N,\epsilon}$ appropriately as a certain large enough polynomial in $\log N$ and ϵ^{-1}, then we can show that the following two distributions are at most $\epsilon/2$-far (in terms of statistical distance):

$$\mathcal{D}_S = \{t(x) \ (\mod \phi(N)) \ : \ x \leftarrow S\},$$
$$\widetilde{\mathcal{D}}_{S,\kappa_{N,\epsilon}} = \left\{\mathsf{CRT}(U_{\mathbb{Z}_{r_1}}, \ldots, U_{\mathbb{Z}_{r_{\kappa_{N,\epsilon}}}}, t(x) \bmod r_{\kappa_{N,\epsilon}+1}, \ldots, t(x) \bmod r_m) \ : \ x \leftarrow S\right\},$$

where $\kappa_{N,\epsilon}$ is such that $r_{\kappa_{N,\epsilon}}$ is smallest prime larger than $\tau_{N,\epsilon}$ and CRT represents the chinese remainder theorem representation.

In the second part of the proof, we show that by relying on Φ-hiding assumption, we can argue that the hash function $t(\cdot)$ can be made lossy on all prime factors of $\phi(N)$ less than or equal to $\tau_{N,\epsilon}$. Combining these two statistical and computational arguments, the smoothness property follows. We point out that throughout this paper, we need to use the aforementioned number theoretic techniques (which involve non-trivial applications of the Prime Number Theorem for Arithmetic Progressions [35]) at multiple places, thus we abstract out and prove many useful number-theoretic lemmas and theorem separately in Chapter 4.

Comparing with DDH-based constructions. Comparing the asymptotic efficiency of our Φ-Hiding based OWFE construction with the existing

DDH-based constructions, we observe the following: (1) the size of the public parameters grows linearly with the input length n in both constructions, (2) both OWF evaluation and decryption operations require $O(n)$ group operations and $O(n)$ exponentiations (with λ-bit exponents) respectively, (3) for the Φ-hiding based construction, both E_1 and E_2 algorithms perform single exponentiation, and outputs a ciphertext and key containing just one group element; whereas for DDH-based construction, the E_1 algorithm performs $O(n)$ exponentiations and outputs a ciphertext containing $O(n)$ group elements.

We implemented the above construction and observed that, at 128-bit security level, our Φ-hiding based construction has $\sim$80x shorter ciphertext size over the existing DDH-based construction [46]. Also, the E_1 algorithm of our Φ-hiding-based construction is $\sim$14x faster than the DDH baseline. We discuss a detailed efficiency comparison for other security levels in Chapter 6.

3.2 Hinting PRGs from Φ-Hiding Assumption

To show general applicability of our *accumulation-style* framework, we also provide a hinting PRG [75] construction based on Φ-hiding that leads to similar performance trade-offs. Let us briefly recall the notion of hinting PRGs. It consists of two algorithms — Setup and Eval, where the setup algorithm generates the public parameters pp, and the PRG evaluation algorithm takes as input the parameters pp, a seed $s \in \{0,1\}^n$ and a block index $i \in \{0, 1, \ldots, n\}$. The security requirement from hinting PRGs is different from standard PRG indistinguishability wherein the hinting PRG scheme is secure if for a randomly

choosen seed $s \in \{0,1\}^n$, the following two distributions over $\{r_{i,b}\}_{(i,b)\in[n]\times\{0,1\}}$ are indistinguishable: in the first distribution, $r_{i,s_i} = \mathsf{Eval}(\mathsf{pp}, s, i)$ and $r_{i,1-s_i}$ is sampled uniformly at random for every i; whereas in the second distribution, all $r_{i,b}$ terms are sampled uniformly at random.

Our hinting PRG construction is based on our OWFE construction, where the setup algorithm is identical, that is the public parameters pp consist of an RSA modulus N, n pairs of λ-bit primes $\{e_{i,b}\}_{(i,b)\in[n]\times\{0,1\}}$, a generator $g \in \mathbb{Z}_N^*$, and a pairwise independent hash H. And, the evaluation algorithm also bears strong resemblance with the one-way function f described previously. Concretely, the i^{*th} block of the PRG output, i.e. $\mathsf{Eval}(\mathsf{pp}, s, i^*)$, is computed as $g^{H(s)\cdot\prod_{i\neq i^*} e_{i,s_i}}(\mod N)$. Proving security of this construction uses ideas similar to those used for proving smoothness property of our OWFE construction. More details on this are provided later in Section 5.4.

Comparing with DDH-based constructions. Comparing the asymptotic efficiency of our Φ-Hiding based hinting PRG construction with the existing DDH-based constructions, we observe the following. (1) the public parameters consist of $2n$ (λ-bit) prime exponents along with the RSA modulus, extractor seed, group generator, and a hash key; whereas in the DDH-based constructions, it contains $O(n^2)$ group elements. (2) for evaluating a single hinting PRG block, the evaluator needs to perform $O(n)$ exponentiations in our new construction. In contrast, in the DDH case, it performs $O(n)$ group operations. Additionally, we reduce the number of exponentiation operations

needed per block to grow only logarithmically in n by using an elegant Dynamic Programming style algorithm (described in Section 5.4.1). The intuition behind such an improvement is that we show how to re-use various intermediate exponentiations obtained during a single hinting PRG block evaluation for accelerating the PRG evaluation for other blocks.

Limitations of Φ-Hiding based constructions. A glance at the above constructions gives us that these new constructions lead to much shorter ciphertext size (in the case of OWFE) and public parameters (in the case of hinting PRGs). Therefore they will lead to better parameters in their corresponding applications such as deterministic encryption [44] and chosen-ciphertext security transformations [75]. However, looking more closely, we observe that our Φ-hiding-based construction has an undesirable consequence. The hinting PRG seed length (or equivalently input length for OWF) n is much larger for our Φ-hiding-based scheme when compared with its DDH counterpart. This is because of the number-field sieve attacks. The recommended RSA modulus length (and thereby the input/seed length n) increases super linearly with the target security level for the Φ-based construction. In comparison, the recommended field size (and thereby the input/seed length n) will increase only linearly for the elliptic curve DDH-based constructions.

Luckily, many prior works [84, 23, 3, 27] studied the notion of accumulators in prime order group setting as well. This gives us a different type of

number theoretic accumulator. Pivoting to such accumulators, we achieve performance improvements similar to that in the Φ-hiding setting while keeping the input/seed length n close to their existing counterparts. Next, we provide our OWFE construction, which uses bilinear maps in the prime order group setting.

3.3 OWF with Encryption from DBDHI Assumption

Let us start by recalling the Decisional Bilinear Diffie-Hellman Inversion (DBDHI) assumption [13]. We characterize the strength of the assumption by a parameter ℓ. The assumption states that given a sequence of group elements as follows — $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{\ell}})$, where g is a random group generator, and α is a randomly chosen non-zero exponent, no PPT adversary should be able to distinguish $e(g,g)^{1/\alpha}$ from a random element in the target group. Below we describe our OWFE construction. Here, we directly include the sequence of elements described above as part of the public parameters.

Concretely, the public parameters pp consist of $n+1$ group elements $(g, g^{\alpha}, \ldots, g^{\alpha^n})$ for a random exponent α and group generator g, and a pairwise independent hash H. (Here n is the input length.) Given an input $x \in \{0,1\}^n$, the one-way function $f_{\mathsf{pp}}(x)$ is computed in two stages. First, the evaluator symbolically evaluates (i.e. simplifies) the polynomial $p(z) = H(x) \cdot \prod_i (z + 2i + x_i)$. Let $p(z) = \sum_{j=0}^{n} c_j z^j$ be the evaluated polynomial. Next, the evaluator sets the output of the OWF as $\prod_j (g^{\alpha^j})^{c_j}$. The encryption algorithm E_1 on input a pre-image bit (i^*, b^*) and randomness ρ, outputs ciphertext as

$\mathsf{ct} = (g^{\alpha+2i^*+b^*})^{\rho}$.[3] The corresponding KEM key is set as $k = e(g^{\rho}, y)$, where y is the output of the OWF. Lastly, the decryption procedure given a ciphertext ct and a pre-image x such that $f_{\mathsf{pp}}(x) = y$ and $x_{i^*} = b^*$, also takes a two step approach where first it symbolically evaluates the polynomial $p'(z) = H(x) \cdot \prod_{i \neq i^*}(z + 2i + x_i)$. Let $p'(z) = \sum_{j=0}^{n-1} c'_j z^j$ be the evaluated polynomial. Lastly, the decryptor computes the key as $k' = e(\mathsf{ct}, \prod_j (g^{\alpha^j})^{c'_j})$.

One-wayness Property. The proof of one-wayness is similar to that in the case of Φ-hiding. Suppose an adversary finds a collision $x \neq x'$, i.e., $f_{\mathsf{pp}}(x) = f_{\mathsf{pp}}(x')$. A reduction algorithm can then set the public parameters appropriately. As long as n is large enough, the reduction algorithm can not only distinguish the DBDHI challenge but also directly compute the DBDHI challenge.

Security of Encryption. The proof of encryption security is also quite similar. We describe the main idea as follows: the ciphertext looks like $\mathsf{ct} = (g^{\alpha+2i^*+b^*})^{\rho}$ whereas the key is computed as $k = e(g^{\rho}, \prod_j (g^{\alpha^j})^{c_j})$. Whenever $b^* \neq x_{i^*}$, then the key can be re-written such that it is of the form $k = e(g, g)^{c'/\beta} \cdot e(g, \prod_j (g^{\beta^j})^{c'_j})$ for some constants $c', c'_1, \ldots, c'_{n-1}$, and where β linearly depends on α. By careful analysis, we can reduce this to the DBDHI assumption.

Smoothness. The proof of smoothness for this construction is significantly simpler than that of its Φ-hiding based counterpart. This is primarily

[3] Technically, the ciphertext should also include the index i^* but we drop it for ease of exposition.

because, in this case, we can directly prove that the function $H(x) \cdot \prod_i (\alpha + 2i + x_i) (\mod p)$, where p is the order of the group, is an (almost) 2-universal hash function, therefore by applying LHL we can argue smoothness of the OWF. More details are provided later in Section 5.3.

Comparing with other constructions. We implemented the above construction and observed that, at 128-bit security level, our DBDHI-based construction has ∼340x shorter ciphertext size over the existing DDH-based construction [46] and ∼4x over our Φ-hiding-based construction. Also, the E_1 algorithm of our DBDHI-based construction is ∼300x faster than the DDH baseline and ∼22x faster than our Φ-hiding construction. Note that even though Φ-hiding and DBDHI-based constructions have nearly identical asymptotic complexity, DBDHI-based construction still perform better as the recommended group size for elliptic curve groups are smaller than that for RSA.

3.4 Hinting PRGs from DDHI and OWFE without Bilinear Maps

To further emphasize the general applicability of our *accumulation-style* framework, we provide a hinting PRG construction based on the DDHI assumption. The translation from OWFE to hinting PRG is done analogous to that for Φ-hiding based constructions, except in our hinting PRG construction, we do not require the bilinear map functionality. Briefly, this is because (unlike OWFE schemes) hinting PRGs do not provide any decryption-like

functionality. For evaluating the hinting PRG, standard group operations are sufficient. We describe our construction in detail later in Section 5.5. We also point out that in Section 5.2 we provide an OWFE construction in the prime order group setting without using bilinear maps. However, the caveat is that it does not lead to better performance when compared with existing DDH-based constructions.

We implemented the above schemes and observed that, at the 128-bit security level, the setup algorithm of our Φ-hiding and DDHI-based HPRGs are $\sim$1.35x and $\sim$200x respectively faster than the DDH baseline [75]. Our constructions also have $\sim$ 105x and $\sim$2100x shorter public parameters, respectively, than the DDH baseline. However, our schemes have a less efficient Eval algorithm. They thereby offer a noticeable trade-off between the efficiency of Setup and Enc algorithm when used in chosen-ciphertext security transformation of [75]. We provide more details later in Chapter 6.

3.5 Additional Related Work

One of our work's applications is constructing trapdoor functions (TDFs) with smaller parameter sizes. Building on the work of [46], Garg, Gay, and Hajiabadi [44] show how OWF with encryption gives trapdoor functions with image size linear in the input size. However, their construction requires a quadratic number of group elements. Plugging in either our bilinear map or ϕ-hiding constructions will reduce the public parameter size to $O(n)$ group elements. (Since our OWFE schemes also satisfy the smoothness criteria, the

resulting TDF also leads to a construction of deterministic encryption.)

In a concurrent work, Garg, Hajiabadi, and Ostrovsky [49] using different techniques give new constructions for "trapdoor hash functions" [41] with small public key size. Among other applications, this also gives an injective trapdoor function whose public key contains $O(n)$ group elements. They prove security from the q-power DDH assumption and use other ideas to reduce the evaluation time. From bilinear maps, however, the work of Boyen and Waters [18] provides TDF constructions secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in which the public keys also have only $O(n)$ group elements.

One interpretation is that the primitive of OWF with encryption can perhaps serve a broader range of applications. However, to squeeze out better performance for a particular more narrow set of applications, a more specialized abstraction such as trapdoor hash functions might be more useful. This mirrors our experience with hinting PRGs, where our direct constructions had efficiency benefits. Finally, we emphasize that part of our contribution is to provide concrete experimental performance measurements of our constructions.

Chapter 4

A Number-Theoretic Toolkit

In this chapter, we build a novel number-theoretic toolkit. This toolkit will be crucial in proving the security of our Φ-hiding based constructions later in Sections 5.1 and 5.4. We will first review the prime number theorem for arithmetic progressions. Even though this theorem is quite famous in the math community, to the best of our knowledge, this is the first time the theorem found a great application in cryptography. We then propose a novel lemma, which we call universal hashing lemma. This lemma shares a resemblance with the left-over hash lemma. Specifically, given a hash function that "close to" to the universal hash function, we show that the hash function's output has high entropy if the input of the hash function is chosen from a high entropy distribution. Unlike left-over hash lemma, which is a purely information-theoretic argument, we use a combination of information-theoretic and computational arguments (based on Φ-hiding assumption) to prove this lemma. Finally, we prove a lemma related to randomness extractors. Throughout this chapter, we consider special groups defined w.r.t. an RSA modulus N.

4.1 Prime Number Theorems for Arithmetic Progressions

First, we recall some important theorems from the number theory literature about prime numbers that we will be relying on in this thesis. In 1837, Dirichlet [35] proved that for two co-prime positive integers a and q, the sequence $\{a + qn\}_{n=0}^{\infty}$ contains infinitely many primes. Further, Dirichlet and Legendre conjectured that the number of primes in this sequence less than x is around $\frac{1}{\phi(q)}\mathsf{Li}(x)$, where $\mathsf{Li}(\cdot)$ is the logarithmic integral function (and a good approximation of the prime counting function). In 1896, de la Vallée Poussin [33] proved the conjecture. Below we state the refined theorem statement as improved in a long line of works (to cite a few [33, 83, 103, 97, 96]). Let PRIMES be the set of all primes numbers, and PRIMES(i) be the set of all i-bit prime numbers.

Theorem 4.1.1 (Prime Number Theorem for Arithmetic Progressions (Paraphrased))**.** *For any two co-prime integers q, a. Define $\theta(x; q, a)$ to be number of primes p less than x such that $a = p \pmod q$. Concretely,*

$$\theta(x; q, a) = \left| \left\{ p \in \mathbb{Z} \;\middle|\; p < x \;\wedge\; p \in \text{PRIMES} \;\wedge\; a = p \pmod q \right\} \right|.$$

Then, the following is true:

$$\forall x, \quad \theta(x; q, a) = (1 + o_q(1))\frac{1}{\phi(q)}\frac{x}{\log x},$$

where the subscript q in the notation $o_q(1)$ denotes that the implied constant could depend q. And, $o_q(1) \to 0$ as $x \to \infty$.

Combining the above theorem with the famed prime number (density) theorem, we get the following corollary.

Corollary 4.1.1 (Prime Number Density in Arithmetic Progressions)**.** *For any two co-prime integers q, a, we have the following.*

$$\forall x, \quad \Pr_{p \leftarrow \{t \in \mathbb{Z} \mid t < x \ \wedge \ t \in \text{PRIMES}\}} [a = p \ (\mathsf{mod}\ q)] = (1 + o_{q,x}(1)) \frac{1}{\phi(q)},$$

where the subscripts q, x in the notation $o_{q,x}(1)$ denotes that the implied constant could depend q, x. And, $o_{q,x}(1) \to 0$ as $x \to \infty$.

In this thesis, we need a slightly stronger guarantee for our results in which the primes p we consider are in a specific range which is fixed bit length. Below we state the corollary, which again follows by combining Theorem 4.1.1 and prime number (density) theorem, and in turn, suffices for our results.

Corollary 4.1.2 (Bounded Range Prime Number Density in Arithmetic Progressions)**.** *For any two co-prime integers q, a, we have the following.*

$$\forall \lambda \in \mathbb{N}, \quad \Pr_{p \leftarrow \text{PRIMES}(\lambda)} [a = p \ (\mathsf{mod}\ q)] = (1 + o_{q,\lambda}(1)) \frac{1}{\phi(q)},$$

where the subscripts q, λ in the notation $o_{q,\lambda}(1)$ denotes that the implied constant could depend q, λ. And, $o_{q,\lambda}(1) \to 0$ as $\lambda \to \infty$.

Remark 4.1.1. *It turns out we need a much weaker guarantee than what is provided above. Concretely, any upper bound as long as it is a fixed inverse polynomial in $\phi(q)$ is sufficient for us.*

4.2 A New Hashing Lemma

Consider an RSA modulus $N = pq$ for $\kappa/2$-bit primes p, q, and let $g \in \mathbb{Z}_N^*$ be a random element in the multiplicative group $\mathbb{Z}_N^*$. Consider the following family of hash functions which hash an n-bit string x ($x \in \mathcal{X} = \{0,1\}^n$) to an element in $\mathbb{Z}_N$:

$$\mathcal{K} = \left\{ (a, b, \{e_{i,c}\}_{i\in[n],b\in\{0,1\}}) \in \mathbb{Z}_N^{2n+2} \ : \ \begin{array}{c} a, b \in \mathbb{Z}_N; \\ e_{i,c} \in \text{PRIMES}(\lambda) \ \forall \ i \in [n], c \in \{0,1\} \end{array} \right\},$$

$$H \ : \ \mathcal{K} \times \mathcal{X} \to \mathbb{Z}_N, \qquad H\left((a, b, \{e_{i,c}\}_{i,c}), x\right) = g^{(ax+b)\prod_i e_{i,x_i}} \ (\mathsf{mod}\ N).$$

Here x is intepreted as an integer for arithmetic operations, and x_i denotes the i^{th} bit of x when intepreted as a binary string. Whenever it is clear from context, we will drop the hash key as an explicit input to the function and write either $H(x)$ or $H_K(x)$ instead of $H(K, x)$ for some hash key $K = (a, b, \{e_{i,c}\}_{i,c})$. Also, throughout we assume that n is sufficiently large, i.e. $n > \kappa + 2\lambda$.

Consider any integer T, and let $T = \prod_{i=1}^{t} r_i^{k_i}$ be its prime factorization i.e., $k_i \geq 1$ and r_i's are the distinct prime factors arranged in an increasing order. For any integer $y \in \mathbb{Z}_T$, we define its chinese remainder theorem (CRT) representation to be the vector $(y^{(1)}, y^{(2)}, \cdots, y^{(t)})$, where for each $i \in [t]$, $y^{(i)} = y \mod r_i^{k_i}$. Note that each integer $y \in \mathbb{Z}_T$ has distinct CRT representation.

Looking ahead, we use the hash function described above in our HPRG and OWFE constructions based on the Φ-hiding assumption. For security, we require that (for a randomly chosen key K and input $x \leftarrow \mathcal{X}$) the output distribution of the hash function $H(K, x)$ to be indistinguishable from a distribution with a large enough min-entropy while looking independent of the input

x. A natural idea would be to use a variant of Leftover Hash Lemma (LHL) to prove such a statement, but since $e_{i,c}$'s are randomly sampled primes (and not random exponents), thus the distribution of the exponent $(ax+b)\prod_i e_{i,x_i}$ mod $\Phi(N)$ is not well understood. Due to this, we could not rely only on LHL to prove the pseudorandomness of the desired distribution but instead show that the hash function satisfies the following weaker property, which is sufficient for our applications. The technical difficulty here lies in proving that the hash function satisfies this weaker property and utilizing this to prove our HPRG and OWFE constructions' security.

Theorem 4.2.1. *Let p_i denote the i^{th} prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$. And, let f_i denote $p_i^{\widetilde{e_i}}$ for all i.*

Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, non-negligible function $\epsilon(\cdot)$, polynomial $v(\cdot)$, for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and $\epsilon = \epsilon(\lambda) > 1/v(\lambda)$, the following holds:

$$\Pr[\mathsf{Expt\text{-}Hashing}_{\mathcal{A},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}Hashing}_{\mathcal{A},\epsilon}(1) = 1] \leq \epsilon(\lambda)/2,$$

where the experiment Expt-Hashing *is described in Figure 4.1.*

Proof. Let the prime factorization of $\phi(N)$ be $\phi(N) = \prod_i r_i^{k_i}$ for $i = 1$ to ℓ_N, where $k_i \geq 1$, ℓ_N denotes number of distinct prime factors of $\phi(N)$, and r_i's are the distinct prime factors arranged in an increasing order. The proof is divided into two parts. First, we argue that $(ax + b)\prod_i e_{i,x_i}$ mod $r_j^{k_j}$ is statistically close to random over $\mathbb{Z}_{r_j^{k_j}}$ for all prime factors of $\phi(N)$ greater than

$\mathsf{Expt\text{-}Hashing}_{\mathcal{A},\epsilon}(\beta)$

The challenger samples RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$, 2 group elements $a, b \leftarrow \mathbb{Z}_N$ and $2n$ λ-bit primes $e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda)$ for $i \in [n], c \in \{0,1\}$. The challenger sets $K = (a, b, \{e_{i,c}\}_{i,c})$.

The challenger now samples (g, y) depending on bit β in the following way.

— If $\beta = 0$, the challenger samples a generator $g \leftarrow \mathbb{Z}_N^*$ and a bit string $x \leftarrow \mathcal{X}$ and computes $y = H_K(x)^{f_1 \cdot f_2}$.

— If $\beta = 1$, the challenger samples generators $\widetilde{g}, h \leftarrow \mathbb{Z}_N^*$. It then sets j_ϵ to be the smallest index such that $p_{j_\epsilon} > (2\sqrt{2}\log N/\epsilon)^3$ and computes $g = \widetilde{g}^{\prod_{i=3}^{j_\epsilon} f_i}$ and $y = h^{\prod_{i=1}^{j_\epsilon} f_i}$.

The challenger sends (N, g, K, y) to the adversary. The adversary then outputs a bit β', and the output of the experiment is set to be the same bit β'.

Figure 4.1: Security experiment for Hashing Lemma

p_{j_ϵ}. In the second part of the proof, we show using Φ-hiding that the hash function H could be made lossy on all prime factors of $\phi(N)$ less than or equal to p_{j_ϵ}. Thus, the theorem follows. For proving the first part, we employ a tight Leftover Hash Lemma proof. And for the second part, we rely on Φ-hiding to introduce lossiness.

Notation. Here and throughout, for any n-bit string x, we use $\mathbf{e}_x$ to denote the following product $\prod_{i \in [n]} e_{i,x_i}$.

Part 1. The statistical argument. Here we show that if we look at the congruent CRT representation of the exponent $(ax + b) \cdot \mathbf{e}_x$ corresponding to prime factors greater p_{j_ϵ}, then (for a randomly chosen hash key K and input x)

they are at most $\epsilon/3$-statistically far from an integer that is chosen at random with the constraint that its congruent CRT representation corresponding to prime factors less than or equal to p_{j_ϵ} is same as for $(ax+b)\cdot \mathbf{e}_x$. Concretely, we show that following:

Lemma 4.2.1. *Let p_i denote the i^{th} prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$.*

For every (possibly unbounded) adversary $\mathcal{A}$, non-negligible function $\epsilon(\cdot)$, polynomial $v(\cdot)$, for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and $\epsilon = \epsilon(\lambda) > 1/v(\lambda)$, the following holds:

$$\Pr[\mathsf{Expt\text{-}NewLHL}_{\mathcal{A},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}NewLHL}_{\mathcal{A},\epsilon}(1) = 1] \leq \epsilon(\lambda)/3,$$

where the experiment $\mathsf{Expt\text{-}NewLHL}_{\mathcal{A},\epsilon}$ is described in Figure 4.2.

Proof. Let $\epsilon = \epsilon(\lambda)$, and the event bad correspond to the scenario when at least one of the λ-bit primes $\{e_{i,c}\}_{i,c}$ are *not* co-prime w.r.t. $\phi(N)$, or $a, b \geq \phi(N)$. Note that the probability of this event happening can be bounded as $\Pr[\mathsf{bad}] \leq 2n \cdot \frac{4\ell_N \lambda}{2^\lambda} + 2 \cdot \frac{\phi(N)}{N} = \mathrm{negl}(\lambda)$. Now for the rest of the analysis, we condition on the event 'bad' not happening, but do not explicitly write it for ease of exposition. That is, in the remaining proof of this theorem we always assume that $\{e_{i,c}\}_{i,c}$ are co-prime w.r.t. $\phi(N)$, and $a, b < \phi(N)$.

Next, consider the following simpler case. For any prime r and exponent k, consider the hash function $h(x) = (ax+b)\prod_i e_{i,x_i} \pmod{r^k}$. Here a, b are sampled uniformly at random from $\mathbb{Z}_{r^k}$ and $e_{i,c}$'s are random λ-bit primes. We first claim the following:

$\mathsf{Expt\text{-}NewLHL}_{\mathcal{A},\epsilon}(\beta)$

The challenger samples RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$, 2 group elements $a, b \leftarrow \mathbb{Z}_N$ and $2n$ λ-bit primes $e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda)$ for $i \in [n], c \in \{0,1\}$. It then samples a bit string $x \leftarrow \mathcal{X}$, sets $K = (a, b, \{e_{i,c}\}_{i,c})$.

The challenger now computes y depending on challenge bit β in the following way.

- If $\beta = 0$, the challenger sets $y = (ax + b) \cdot \mathbf{e}_x \pmod{\phi(N)}$.
- If $\beta = 1$,
  - Let the prime factorization of $\phi(N)$ be $\phi(N) = \prod r_i^{k_i}$, where $k_i \geq 1$, and r_i's are the distinct prime factors arranged in an increasing order. Let ℓ_N denotes number of distinct prime factors of $\phi(N)$.
  - It then sets $\widetilde{y} = (ax + b) \cdot \mathbf{e}_x \pmod{\phi(N)}$ and computes its CRT representation $\widetilde{y} = (\widetilde{y}^{(1)}, \ldots, \widetilde{y}^{(\ell_N)})$, where $\widetilde{y}^{(i)} = \widetilde{y} \pmod{r_i^{k_i}}$.
  - The challenger then sets j_ϵ to be the smallest index such that $p_{j_\epsilon} > (2\sqrt{2}\log N/\epsilon)^3$. For each for $i \in [\ell_N]$ such that $r_i \leq p_{j_\epsilon}$, the challenger sets $y^{(i)} = \widetilde{y}^{(i)}$. For each $i \in [\ell_N]$ such that $r_i > p_{j_\epsilon}$, it samples $y^{(i)} \leftarrow \mathbb{Z}_{r_i^{k_i}}$.
  - The challenger then computes y which has CRT representation $(y^{(1)}, \ldots, y^{(\ell_N)})$.

The challenger sends (N, K, y) to the adversary. The adversary then outputs a bit $\beta' \in \{0,1\}$, and the output of the experiment is set to be the same bit β'.

Figure 4.2: Security Game for Lemma 4.2.1

Claim 4.2.1. *For every prime $r > 3$, integer $k \geq 1$,*

$$\Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x \neq y}} \left[(ax+b)\cdot \mathbf{e}_x = (ay+b)\cdot \mathbf{e}_y \pmod{r^k}\right] \leq \frac{1}{r^k}\left(1 + \frac{2k}{r^{2/3}} + \frac{k \cdot r^k}{|\mathcal{X}|}\right).$$

Proof. Note that we have already conditioned on the event that all the λ-bit primes $\{e_{i,c}\}_{i,c}$ are co-prime w.r.t. $\phi(N)$. (This happens with all but negligible

probability, thus does not affect the remaining analysis.) Now fix any $c \in \mathbb{Z}_{r^k}$. We have that if

$$(ax+b)\cdot \mathbf{e}_x = (ay+b)\cdot \mathbf{e}_y = c \implies \begin{matrix} ax+b = c\cdot \mathbf{e}_x^{-1} \\ ay+b = c\cdot \mathbf{e}_y^{-1} \end{matrix} \implies a(x-y) = c\left(\mathbf{e}_x^{-1} - \mathbf{e}_y^{-1}\right).$$

Consider the following cases — (1) $r \nmid x-y$, (2) $r \mid x-y \ \wedge \ r^2 \nmid x-y$, (3) ..., (k) $r^{k-1} \mid x-y \ \wedge \ r^k \nmid x-y$, $(k+1)$ $r^k \mid x-y$. We know that $\Pr_{x\neq y}[\text{Case } (i)] \leq (\frac{1}{r^{i-1}} - \frac{1}{r^i} + \frac{r^i}{|\mathcal{X}|})$ for $i < k+1$ and $\Pr_{x\neq y}[\text{Case } (k+1)] \leq 1/r^k$. Now, in case (1), for every $c \in \mathbb{Z}_{r^k}$, there exists a unique (a,b) pair such that the aforementioned equations are satisfied. (Because if $r \nmid x-y$, then $(x-y)^{-1}$ is unique and always exists.) So, we can write that

$$\Pr_{a,b}\left[(ax+b)\cdot \mathbf{e}_x = (ay+b)\cdot \mathbf{e}_y \ (\mathsf{mod}\ r^k) \ \middle|\ \text{Case } (1)\right] = \frac{1}{r^k}.$$

Next, consider case (i) for $i > 1$. There are $(i-1)$ sub-cases — $(i.1)$ $r \nmid \mathbf{e}_x - \mathbf{e}_y$, $(i.2)$ $r \mid \mathbf{e}_x - \mathbf{e}_y \wedge r^2 \nmid \mathbf{e}_x - \mathbf{e}_y$, (3) ..., $(i.(i-1))$ $r^{i-2} \mid \mathbf{e}_x - \mathbf{e}_y \wedge r^{i-1} \nmid \mathbf{e}_x - \mathbf{e}_y$, $(i.i)$ $r^{i-1} \mid \mathbf{e}_x - \mathbf{e}_y$. In case $(i.1)$, for a collision to occur it must hold that $c = 0 \ \mathsf{mod}\ r^{i-1}$ since $x - y = 0 \ \mathsf{mod}\ r^{i-1}$ (as $r^{i-1} \mid x-y$). We have that the number of such c values is r^{k-i+1}. Now for each such c, we can solve for r^{i-1} pairs of solutions for (a,b). Similarly in other cases, i.e. case $(i.j)$ for $1 < j \leq i$, we have that number of satisfying c values in $\mathbb{Z}_{r^k}$ will be r^{k-i+j}, and for each such c there will exist r^{i-1} pairs of solutions for (a,b). Now, for any i, let $\pi(r^i)$ denote the following probability

$$\pi(r^i) = \Pr_{\{e_{i,c}\}_{i,c}}\left[\mathbf{e}_x = \mathbf{e}_y \ (\mathsf{mod}\ r^i)\right].$$

Next, combining all the above cases and sub-cases, we get the following:

$$\Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x\neq y}}\left[(ax+b)\cdot\mathbf{e}_x=(ay+b)\cdot\mathbf{e}_y\ (\mathsf{mod}\ r^k)\right]\leq\left(1-\frac{1}{r}+\frac{r}{|\mathcal{X}|}\right)\frac{1}{r^k}$$
$$+\sum_{i=2}^{k}\left(\frac{1}{r^{i-1}}-\frac{1}{r^i}+\frac{r^i}{|\mathcal{X}|}\right)\left(r^{k-i+1}\cdot\frac{r^{i-1}}{r^{2k}}+k\cdot\pi(r^{i-1})\cdot r^k\cdot\frac{r^{i-1}}{r^{2k}}\right)$$
$$+\frac{1}{r^k}\left(\frac{r^k}{r^{2k}}+k\cdot\pi(r^k)\cdot r^k\cdot\frac{r^k}{r^{2k}}\right).$$

$$\Rightarrow\Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x\neq y}}\left[(ax+b)\cdot\mathbf{e}_x=(ay+b)\cdot\mathbf{e}_y\ (\mathsf{mod}\ r^k)\right]\tag{4.1}$$
$$\leq\frac{1}{r^k}+\frac{k}{|\mathcal{X}|}+\frac{k}{r^k}\left(1-\frac{1}{r}\right)\sum_{i=2}^{k}\pi(r^{i-1})+\frac{1}{r^k}\cdot\pi(r^k).\tag{4.2}$$
$$\leq\frac{1}{r^k}\left(1+k\cdot\sum_{i=1}^{k}\pi(r^i)\right)+\frac{k}{|\mathcal{X}|}.\tag{4.3}$$

Now let us try to bound the probability $\pi(r^i)$. Fix any $x\neq y$ and let j^* denote the first index such that $x_{j^*}\neq y_{j^*}$. Note that,

$$\pi(r^i)=\Pr_{\{e_{i,c}\}_{i,c}}\left[\mathbf{e}_x=\mathbf{e}_y\ (\mathsf{mod}\ r^i)\right]=\Pr_{e_{j^*,x_{j^*}}}\left[e_{j^*,x_{j^*}}=\mathbf{e}_y\cdot\prod_{j\neq j^*}e_{j,x_j}^{-1}\ (\mathsf{mod}\ r^i)\right].$$

Next, using the theorem on bounded range prime number density in arithmetic progressions (see Corollary 4.1.2), we get that $\pi(r^i)\leq\frac{1}{r^{2i/3}}$.[1] Therefore, we get that

$$\sum_{i=1}^{k}\pi(r^i)\leq\frac{1-r^{-2(k+1)/3}}{r^{2/3}-1}\leq 2r^{-2/3}.$$

[1]Note that here we are using a very weak upper bound. Our analysis could be further tightened, but since a weaker guarantee is sufficient for the proof, thus we stick with it.

Combining this with Equation (4.1), we get that

$$\Rightarrow \Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x\neq y}} \left[(ax+b)\cdot \mathbf{e}_x = (ay+b)\cdot \mathbf{e}_y \pmod{r^k}\right] \leq \frac{1}{r^k}\left(1+\frac{2k}{r^{2/3}}\right)+\frac{k}{|\mathcal{X}|}. \tag{4.4}$$

This completes the proof of Claim 4.2.1. ■

Next, using Claim 4.2.1, we can claim the following.

Claim 4.2.2. *For every prime $r > 3$, integer $k \geq 1$, every (possibly unbounded) adversary $\mathcal{A}$, the following holds*

$$\Pr\left[\mathcal{A}(r^k, K, y_\beta) = \beta \;:\; \begin{array}{c} a,b \leftarrow \mathbb{Z}_{r^k}; e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda) \\ K = (a,b,\{e_{i,c}\}_{i,c}); \quad x \leftarrow \mathcal{X} \\ y_0 = (ax+b)\cdot \mathbf{e}_x \pmod{r^k}; \\ y_1 \leftarrow \mathbb{Z}_{r^k} \end{array}\right] \leq \sqrt{\frac{k}{2r^{2/3}}}+\sqrt{\frac{kr^k}{2|\mathcal{X}|}}$$

Proof. The proof of this lemma is similar to the proof of Leftover Hash Lemma [66, 37, 36] where the collision probability is used as obtained in Claim 4.2.1. Concretely, for any prime r and exponent k, consider the hash function $H^{(r,k)}(K^{(r,k)}, x) = (ax+b)\prod_i e_{i,x_i} \pmod{r^k}$. Here a, b are sampled uniformly at random from $\mathbb{Z}_{r^k}$ and $e_{i,c}$'s are random λ-bit primes. The key consists of $K^{(r,k)} = (a, b, \{e_{i,c}\}_{i,c})$. Note that hash function $H^{(r,k)} : \mathcal{K}^{(r,k)} \times \mathcal{X} \to \mathbb{Z}_{r^k}$.

Let $\delta = \Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x\neq y}} \left[(ax+b)\cdot \mathbf{e}_x = (ay+b)\cdot \mathbf{e}_y \pmod{r^k}\right]$. Now, we can

write the following

$$\begin{aligned}
&\mathsf{SD}\left((H^{(r,k)}, H^{(r,k)}(X)), (H^{(r,k)}, U_{\mathbb{Z}_{r^k}})\right) \\
&\quad \leq \frac{1}{2}\sqrt{|\mathcal{K}^{(r,k)}| \cdot r^k}\sqrt{\frac{\delta}{|\mathcal{K}^{(r,k)}|} + \frac{1}{|\mathcal{X}| \cdot |\mathcal{K}^{(r,k)}|} - \frac{1}{|\mathcal{K}^{(r,k)}| \cdot r^k}} \\
&\quad \leq \frac{1}{2}\sqrt{\delta \cdot r^k - 1 + \frac{r^k}{|\mathcal{X}|}} \\
&\quad \leq \frac{1}{2}\left(\sqrt{\delta \cdot r^k - 1} + \sqrt{\frac{r^k}{|\mathcal{X}|}}\right)
\end{aligned}$$

where SD corresponds to the statistical distance. Using Claim 4.2.1, we can simply it further to $\sqrt{\frac{k}{2r^{2/3}}} + \sqrt{\frac{kr^k}{2|\mathcal{X}|}}$. This completes the proof of Claim 4.2.2. ■

Finally, using union bounds, congruence due to Chinese remainder theorem, and extending the analyses of Claim 4.2.1 and Claim 4.2.2, we get the following

$$\begin{aligned}
&\Pr[\mathsf{Expt\text{-}NewLHL}_{\mathcal{A},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}NewLHL}_{\mathcal{A},\epsilon}(1) = 1] \\
&\qquad \leq \Pr[\mathsf{Bad}] + \frac{\mathsf{fac}(N)}{\sqrt{2p_{j_\epsilon}^{2/3}}} + \sqrt{\frac{\log N \cdot \phi(N)}{2|\mathcal{X}|}} \\
&\qquad \leq \mathrm{negl}_1(\lambda) + \epsilon/4 + \mathrm{negl}_2(\lambda) < \epsilon/3.
\end{aligned}$$

Here $\mathsf{fac}(N)$ is the total number of prime factors of N (where factors with multiplicity > 1 are counted multiple times) which is bounded by $\log N$. Here to argue that $\sqrt{\frac{\log N \cdot \phi(N)}{2|\mathcal{X}|}}$ is negligible in λ, we use the fact that input domain $\mathcal{X} = \{0,1\}^n$ is quite large, i.e. $n > \kappa + 2\lambda$.

This completes the proof of the first part, which shows a tight bound on the statistical distance between the real and intermediate hybrid distribution. ■

Part 2. The computational argument. Here we show that, using Φ-hiding, the generator g instead of sampling uniformly at random could be sampled as $g^{\prod_{i=1}^{j_\epsilon} f_i}$, where j_ϵ is the smallest index such that $p_{j_\epsilon} > (2\sqrt{2}\log N/\epsilon)^3$. This removes information about the input x completely. Concretely, we show that following:

Lemma 4.2.2. *Let p_i denote the i^{th} prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$ and let f_i denote $p_i^{\widetilde{e_i}}$ for all i. Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, non-negligible function $\epsilon(\cdot)$, polynomial $v(\cdot)$, for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and $\epsilon = \epsilon(\lambda) > 1/v(\lambda)$, there exists a negligible function $negl(\cdot)$ such that the following holds,*

$$\Pr[\mathsf{Expt\text{-}Comp}_{\mathcal{A},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}Comp}_{\mathcal{A},\epsilon}(1) = 1] \leq negl(\lambda),$$

where the experiment $\mathsf{Expt\text{-}Comp}_{\mathcal{A},\epsilon}$ is described in Figure 4.3.

Proof. The proof proceeds by a sequence of $j_\epsilon - 2$ hybrids. For each $2 \leq i^* \leq j_\epsilon$, let us define intermediate hybrid experiment $\mathsf{Hybrid\text{-}Comp}(i^*)$ be same as $\mathsf{Expt\text{-}Comp}(0)$ except that the challenger computes (g, h) in the following way: The challenger samples generator $\tilde{g} \leftarrow \mathbb{Z}_N^*$ and sets $g = \tilde{g}^{\prod_{i=3}^{i^*} f_i}$ and $h = g^{y \cdot f_1 \cdot f_2}$.

Formally, we prove the following under the Φ-hiding assumption:

$\mathsf{Expt\text{-}Comp}_{\mathcal{A},\epsilon}(\beta)$

The challenger samples RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$, 2 group elements $a, b \leftarrow \mathbb{Z}_N$ and $2n$ λ-bit primes $e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda)$ for $i \in [n], c \in \{0,1\}$, and sets $K = (a, b, \{e_{i,c}\}_{i,c})$. It then samples a bit string $x \leftarrow \mathcal{X}$.

The challenger then sets j_ϵ to be the smallest index such that $p_{j_\epsilon} > (2\sqrt{2}\log N/\epsilon)^3$.

The challenger now computes (g, h) depending on challenge bit β in the following way.

- If $\beta = 0$,
  - Let the prime factorization of $\phi(N)$ be $\phi(N) = \prod r_i^{k_i}$, where $k_i \geq 1$, and r_i's are the distinct prime factors arranged in an increasing order. Let ℓ_N denotes number of distinct prime factors of $\phi(N)$.
  - It then sets $\widetilde{y} = (ax + b) \cdot \mathbf{e}_x \pmod{\phi(N)}$.
  - For each for $i \in [\ell_N]$ such that $r_i \leq p_{j_\epsilon}$, the challenger sets $y^{(i)} = \widetilde{y} \pmod{r_i^{k_i}}$. For each $i \in [\ell_N]$ such that $r_i > p_{j_\epsilon}$, it samples $y^{(i)} \leftarrow \mathbb{Z}_{r_i^{k_i}}$. The challenger then computes y which has CRT representation $(y^{(1)}, \ldots, y^{(\ell_N)})$.
  - It then samples a generator $g \leftarrow \mathbb{Z}_N^*$ and sets $h = g^{y \cdot f_1 \cdot f_2}$.
- If $\beta = 1$, the challenger samples generators $\tilde{g}, \tilde{h} \leftarrow \mathbb{Z}_N^*$ and sets $g = \tilde{g}^{\prod_{i=3}^{j_\epsilon} f_i}$, $h = \tilde{h}^{\prod_{i=1}^{j_\epsilon} f_i}$.

The challenger sends (N, g, K, h) to the adversary. The adversary then outputs a bit $\beta' \in \{0,1\}$, and the output of the experiment is set to be the same bit β'.

Figure 4.3: Security Game for Lemma 4.2.2

Claim 4.2.3. *Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and every index $2 \leq i^* < j_\epsilon$, the following holds:*

$$\Pr[\mathsf{Hybrid\text{-}Comp}(i^*) = 1] - \Pr[\mathsf{Hybrid\text{-}Comp}(i^* + 1) = 1] \leq negl(\lambda).$$

Proof. Suppose there is a PPT adversary $\mathcal{A}$ which can distinguish between the two hybrid distributions with non-negligible probability γ. We use $\mathcal{A}$ to construct a reduction algorithm $\mathcal{B}$ that breaks the Φ-hiding assumption for the $(i^*+1)^{th}$ prime p_{i^*+1} with advantage negligibly close to γ.

The Φ-hiding challenger samples an RSA modulus N and sends to reduction algorithm $\mathcal{B}$. The reduction algorithm samples the key K by choosing parameters a, b, primes $\{e_{i,c}\}_{i,c}$ as in the hybrid distributions. It also chooses a random input $x \leftarrow \mathcal{X}$. It computes $\widetilde{y} = (ax+b)\mathbf{e}_x$. (Here computation is done in an absolute sense, not as modular arithmetic.) Next, $\mathcal{B}$ samples a generator $\widetilde{g} \leftarrow \mathbb{Z}_N^*$. (Here it actually samples $\tilde{g} \leftarrow \mathbb{Z}_N$ and aborts whenever $g \notin \mathbb{Z}_N^*$. This happens with only negligible probability.) It computes $g = \widetilde{g}^{\prod_{i=3}^{i^*+1} f_i}$. Next, it computes h as $h = \widetilde{g}^{\widetilde{y}\prod_{i=1}^{i^*+1} f_i} \times \widetilde{h}^{\prod_{i=1}^{j_\epsilon} f_i}$ where $\widetilde{h} \leftarrow \mathbb{Z}_N^*$. Finally, it sends (N, g, K, h) to the adversary $\mathcal{A}$. If the adversary $\mathcal{A}$ outputs 1, then $\mathcal{B}$ guesses that p_{i^*+1} does not divide $\phi(N)$, else it guesses that $p_{i^*+1} \mid \phi(N)$.

Let us now analyze the advantage of the reduction algorithm $\mathcal{B}$. First, recall that $f_{i^*+1} = p_{i^*+1}^{\tilde{e}_{i^*+1}}$. Now if $p_{i^*+1} \nmid \phi(N)$, then the distributions $\{\widetilde{g} \ : \ \widetilde{g} \leftarrow \mathbb{Z}_N^*\}$ and $\left\{\widetilde{g}^{f_{i^*+1}} \ : \ \widetilde{g} \leftarrow \mathbb{Z}_N^*\right\}$ are identically distributed. Therefore, in this case, the reduction algorithm simulates experiment $\mathsf{Hybrid\text{-}Comp}(i^*)$ for $\mathcal{A}$. Otherwise, if $p_{i^*+1} \mid \phi(N)$, then $\mathcal{B}$ simulates the experiment $\mathsf{Hybrid\text{-}Comp}(i^*+1)$ for $\mathcal{A}$. (Note that in both cases it perfectly simulates the element h as well. This is because note that the multiplicative term $\widetilde{h}^{\prod_{i=1}^{j_\epsilon} f_i}$ for a random choice of $\widetilde{h}$ simply randomizes the congruent CRT representation corresponding to prime factors greater than p_{j_ϵ}. This is exactly what the distributions require, thus

simulation is done perfectly.) Hence, if $\mathcal{A}$ distinguishes with non-negligible probability γ, then $\mathcal{B}$'s advantage in Φ-hiding is negligibly close to γ. Thus, the lemma follows. ■

Lastly, to complete the proof of Lemma 4.2.2, we argue the following:

Claim 4.2.4. *For any PPT adversary $\mathcal{A}$ and non-negligible ϵ,*

$$\Pr[\mathsf{Expt\text{-}Comp}(1) = 1] = \Pr[\mathsf{Hybrid\text{-}Comp}(j_\epsilon) = 1]$$

Note that the only difference in both the experiments is the way the element h is computed. In $\mathsf{Hybrid\text{-}Comp}(j_\epsilon)$, $h = \tilde{g}^{y \prod_{i=1}^{j_\epsilon} f_i}$, whereas in $\mathsf{Expt\text{-}Comp}(1)$, $h = \tilde{h}^{\prod_{i=1}^{j_\epsilon} f_i}$. We know that, $y \pmod{r_i^{k_i}}$ is sampled uniformly for all i such that $r_i > p_{j_\epsilon}$. Therefore, the distribution of $y \cdot \prod_{i=1}^{j_\epsilon} f_i$ is identical to $y' \cdot \prod_{i=1}^{j_\epsilon} f_i$, where $y' \leftarrow \mathbb{Z}_{\phi(N)}$. As the distribution of $\{g^{y'} : y' \leftarrow \mathbb{Z}_{\phi(N)}\}$ is identical to the distribution $\tilde{h} \leftarrow \mathbb{Z}_N^*$, the claim follows.

Finally, combining above Claim 4.2.3 and Claim 4.2.4, the Lemma 4.2.2 follows.

■

Lastly, by combining Lemmas 4.2.1 and 4.2.2, we obtain the proof of Theorem 4.2.1.

■

4.3 Strengthening the Hash Lemma

In this section, we briefly provide a slight strengthening of the Theorem 4.2.1 where we argue that the indistinguishability holds even if the input $x \in \mathcal{X}$, instead of being sampled uniformly at random, is sampled from any arbitrary distribution with certain min-entropy. Formally, we prove the following.

Theorem 4.3.1. *Let p_i denote the i^{th} prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e}_i = \lceil \log_{p_i} N \rceil$. And, let f_i denote $p_i^{\widetilde{e}_i}$ for all i.*

Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, non-negligible function $\epsilon(\cdot)$, polynomial $v(\cdot)$, for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and $\epsilon = \epsilon(\lambda) > 1/v(\lambda)$, and every (m, n)-source $\mathcal{S}$ over $\mathcal{X}$ such that $n - m = O(\log \lambda)$, the following holds,

$$\Pr[\mathsf{Expt\text{-}Hashing\text{-}Smooth}_{\mathcal{A},\mathcal{S},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}Hashing\text{-}Smooth}_{\mathcal{A},\mathcal{S},\epsilon}(1) = 1] \leq \epsilon(\lambda)/2,$$

where the experiment Expt-Hashing-Smooth *is described in Figure 4.4.*

Proof. The proof of the above theorem is nearly identical to the proof of Theorem 4.2.1, where we need to slightly adapt the statistical argument to account for the entropy loss. Here we briefly highlight the modifications necessary for proving the above theorem.

Similar to the proof of Theorem 4.2.1, we first prove the following statistical argument:

$\mathsf{Expt\text{-}Hashing\text{-}Smooth}_{\mathcal{A},\mathcal{S},\epsilon}(\beta)$

The challenger samples RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$, 2 group elements $a, b \leftarrow \mathbb{Z}_N$ and $2n$ λ-bit primes $e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda)$ for $i \in [n], c \in \{0,1\}$. The challenger sets $K = (a, b, \{e_{i,c}\}_{i,c})$.

The challenger now samples (g, y) depending on bit β in the following way.

— If $\beta = 0$, the challenger samples a generator $g \leftarrow \mathbb{Z}_N^*$ and a bit string $x \leftarrow \mathcal{S}$ and computes $y = H_K(x)^{f_1 \cdot f_2}$.

— If $\beta = 1$, the challenger samples generators $\widetilde{g}, h \leftarrow \mathbb{Z}_N^*$. It then sets j_ϵ to be the smallest index such that $p_{j_\epsilon} > (2^{n-m+2} \log N/\epsilon)^3$ and computes $g = \widetilde{g}^{\prod_{i=3}^{j_\epsilon} f_i}$ and $y = h^{\prod_{i=1}^{j_\epsilon} f_i}$.

The challenger sends (N, g, K, y) to the adversary. The adversary then outputs a bit β', and the output of the experiment is set to be the same bit β'.

Figure 4.4: Security experiment for Smooth Hashing Lemma (Theorem 4.3.1)

Lemma 4.3.1. *Let p_i denote the i^{th} prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$.*

For every (possibly unbounded) adversary $\mathcal{A}$, non-negligible function $\epsilon(\cdot)$, polynomial $v(\cdot)$, for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and $\epsilon = \epsilon(\lambda) > 1/v(\lambda)$, and every (m,n)-source $\mathcal{S}$ over $\mathcal{X}$ such that $n - m = O(\log \lambda)$, the following holds,

$$\Pr[\mathsf{Expt\text{-}NewLHL\text{-}Smooth}_{\mathcal{A},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}NewLHL\text{-}Smooth}_{\mathcal{A},\epsilon}(1) = 1] \leq \epsilon(\lambda)/3,$$

where the experiment $\mathsf{Expt\text{-}NewLHL\text{-}Smooth}_{\mathcal{A},\epsilon}$ is described in Figure 4.5.

Proof. The proof of the above theorem is nearly identical to the proof of Lemma 4.2.1, with the following changes. Here, the bad events and all the cases (and sub-cases) are identically defined. The difference is that first, we

$\mathsf{Expt\text{-}NewLHL\text{-}Smooth}_{\mathcal{A},\epsilon}(\beta)$

The challenger samples RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$, 2 group elements $a, b \leftarrow \mathbb{Z}_N$ and $2n$ λ-bit primes $e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda)$ for $i \in [n], c \in \{0,1\}$. It then samples a bit string $x \leftarrow \mathcal{X}$, sets $K = (a, b, \{e_{i,c}\}_{i,c})$.

The challenger now computes y depending on challenge bit β in the following way.

- If $\beta = 0$, the challenger sets $y = (ax+b) \cdot \mathbf{e}_x \pmod{\phi(N)}$.
- If $\beta = 1$,
  - Let the prime factorization of $\phi(N)$ be $\phi(N) = \prod r_i^{k_i}$, where $k_i \geq 1$, and r_i's are the distinct prime factors arranged in an increasing order. Let ℓ_N denotes number of distinct prime factors of $\phi(N)$.
  - It then sets $\widetilde{y} = (ax+b) \cdot \mathbf{e}_x \pmod{\phi(N)}$ and computes its CRT representation $\widetilde{y} = (\widetilde{y}^{(1)}, \ldots, \widetilde{y}^{(\ell_N)})$, where $\widetilde{y}^{(i)} = \widetilde{y} \pmod{r_i^{k_i}}$.
  - The challenger then sets j_ϵ to be the smallest index such that $p_{j_\epsilon} > (2^{n-m+2} \log N/\epsilon)^3$. For each for $i \in [\ell_N]$ such that $r_i \leq p_{j_\epsilon}$, the challenger sets $y^{(i)} = \widetilde{y}^{(i)}$. For each $i \in [\ell_N]$ such that $r_i > p_{j_\epsilon}$, it samples $y^{(i)} \leftarrow \mathbb{Z}_{r_i^{k_i}}$.
  - The challenger then computes y which has CRT representation $(y^{(1)}, \ldots, y^{(\ell_N)})$.

The challenger sends (N, K, y) to the adversary. The adversary then outputs a bit $\beta' \in \{0,1\}$, and the output of the experiment is set to be the same bit β'.

Figure 4.5: Security Game for Lemma 4.3.1

need to prove an alternate version of Claim 4.2.1, where the inputs x, y are now sampled as per $\mathcal{S}$ instead.

Claim 4.3.1. *For every prime $r > 3$, integer $k \geq 1$,*

$$\Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x,y \leftarrow \mathcal{S},\\ x \neq y}} \left[(ax+b) \cdot \mathbf{e}_x = (ay+b) \cdot \mathbf{e}_y \pmod{r^k}\right] \leq \frac{1}{r^k}\left(1 + \frac{2^{n-m+1}k}{r^{2/3}} + \frac{2^{n-m} r^k k}{|\mathcal{X}|}\right).$$

Proof. The proof of this lemma is identical to that of Claim 4.2.1, except the probability that any of the cases (1) to $(k+1)$ occur gets amplified by a multiplicative factor of 2^{n-m}. This follows directly from the fact that the min-entropy of $\mathcal{S}$ is m and the length of inputs is n bits. Concretely, now we will have the following:

$$\begin{aligned} &\Pr_{\substack{x,y\leftarrow\mathcal{S},\\ x\neq y}}[\text{Case } (1)] \leq 1 - \frac{2^{n-m}}{r} + \frac{2^{n-m}r}{|\mathcal{X}|}, \\ \text{For } 1 < i < k+1, \qquad &\Pr_{\substack{x,y\leftarrow\mathcal{S},\\ x\neq y}}[\text{Case } (i)] \leq 2^{n-m}\left(\frac{1}{r^{i-1}} - \frac{1}{r^i} + \frac{r^i}{|\mathcal{X}|}\right), \\ &\Pr_{\substack{x,y\leftarrow\mathcal{S},\\ x\neq y}}[\text{Case } (k+1)] \leq \frac{2^{n-m}}{r^k} \end{aligned}$$

Now the rest of analysis follows analogously where the only difference is that we have to take this extra multiplicative factor of 2^{n-m} into account in every equation. Thus, following the analysis instead of Equation (4.4), we get the following:

$$\Pr_{\substack{a,b,\{e_{i,c}\}_{i,c},\\ x,y\leftarrow\mathcal{S},\\ x\neq y}}\left[(ax+b)\cdot\mathbf{e}_x = (ay+b)\cdot\mathbf{e}_y \pmod{r^k}\right] \leq \frac{1}{r^k}\left(1 + \frac{2^{n-m+1}k}{r^{2/3}}\right) + \frac{2^{n-m}k}{|\mathcal{X}|}. \tag{4.5}$$

This completes the proof of Claim 4.3.1. ∎

Next, using Claim 4.3.1, (as before) we can claim the following.

Claim 4.3.2. *For every prime $r > 3$, integer $k \geq 1$, every (possibly un-*

bounded) adversary $\mathcal{A}$, the following holds

$$\Pr\left[\mathcal{A}(r^k, K, y_\beta) = \beta \;:\; \begin{array}{c} a, b \leftarrow \mathbb{Z}_{r^k}; e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda) \\ K = (a, b, \{e_{i,c}\}_{i,c}); \quad x \leftarrow \mathcal{S} \\ y_0 = (ax+b) \cdot \mathbf{e}_x \ (\mathsf{mod}\ r^k);\ y_1 \leftarrow \mathbb{Z}_{r^k} \end{array}\right] \leq 2^{(n-m)/2}\left(\sqrt{\frac{k}{2r^{2/3}}} + \sqrt{\frac{kr^k}{2|\mathcal{X}|}}\right).$$

Proof. The proof of this lemma is similar to that of Claim 4.2.2. Concretely, when the input x is drawn as $x \leftarrow \mathcal{S}$. We can write the following. Let $\delta = \Pr_{\substack{a,b,\{e_{i,c}\}_{i,c}, \\ x,y\leftarrow\mathcal{S}, \\ x\neq y}}\left[(ax+b)\cdot\mathbf{e}_x = (ay+b)\cdot\mathbf{e}_y \ (\mathsf{mod}\ r^k)\right]$.

$$\begin{aligned}
&\mathsf{SD}\left((H^{(r,k)}, H^{(r,k)}(X)), (H^{(r,k)}, U_{\mathbb{Z}_{r^k}})\right) \\
&\quad\leq \frac{1}{2}\sqrt{|\mathcal{K}^{(r,k)}|\cdot r^k}\sqrt{\frac{\delta}{|\mathcal{K}^{(r,k)}|} + \frac{2^{n-m}}{|\mathcal{X}|\cdot|\mathcal{K}^{(r,k)}|} - \frac{1}{|\mathcal{K}^{(r,k)}|\cdot r^k}} \\
&\quad\leq \frac{1}{2}\sqrt{\delta\cdot r^k - 1 + \frac{2^{n-m}r^k}{|\mathcal{X}|}} \\
&\quad\leq \frac{1}{2}\left(\sqrt{\delta\cdot r^k - 1} + \sqrt{\frac{2^{n-m}r^k}{|\mathcal{X}|}}\right)
\end{aligned}$$

Using Claim 4.3.1, we can simply it further to $\sqrt{\frac{2^{n-m}k}{2r^{2/3}}} + \sqrt{\frac{2^{n-m}r^k k}{2|\mathcal{X}|}}$. This completes the proof of Claim 4.3.2.

■

Finally, using union bounds, congruence due to Chinese remainder theorem, and extending the analyses of Claim 4.3.1 and Claim 4.3.2, we get the following

$$\begin{aligned}
&\Pr[\mathsf{Expt\text{-}NewLHL\text{-}Smooth}_{\mathcal{A},\epsilon}(0) = 1] - \Pr[\mathsf{Expt\text{-}NewLHL\text{-}Smooth}_{\mathcal{A},\epsilon}(1)] \\
&\leq \Pr[\mathsf{Bad}] + \frac{2^{(n-m)/2}\mathsf{fac}(N)}{\sqrt{2p_{j_\epsilon}^{2/3}}} + \sqrt{\frac{2^{n-m}\log N \cdot \phi(N)}{2|\mathcal{X}|}} \\
&\leq \mathrm{negl}_1(\lambda) + \epsilon/4 + \mathrm{negl}_2(\lambda) < \epsilon/3.
\end{aligned}$$

Here $\mathsf{fac}(N)$ is the total number of prime factors of N (where factors with multiplicity > 1 are counted multiple times) which is bounded by $\log N$. Here to argue that $\sqrt{\frac{2^{n-m}\log N \cdot \phi(N)}{2|\mathcal{X}|}}$ is negligible in λ, we use the fact that input domain $\mathcal{X} = \{0,1\}^n$ is quite large, i.e. $n > \kappa + 2\lambda$.

This completes the proof of the first part, which shows a tight bound on the statistical distance between the real and intermediate hybrid distribution. ■

Finally, to complete the proof of Theorem 4.3.1, we prove using the Φ-hiding assumption the computational part of the argument. Concretely, we show that following:

Lemma 4.3.2. *Let p_i denote the i^{th} prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$ and let f_i denote $p_i^{\widetilde{e_i}}$ for all i. Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, non-negligible function $\epsilon(\cdot)$, polynomial $v(\cdot)$, for all $\lambda, \kappa \in \mathbb{N}$, satisfying $\kappa \geq 5\lambda$ and $\epsilon = \epsilon(\lambda) > 1/v(\lambda)$, and every (m,n)-source $\mathcal{S}$ over $\mathcal{X}$ such that $n - m = O(\log \lambda)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that the following holds,*

$$\Pr[\mathsf{Expt\text{-}Comp\text{-}Smooth}_{\mathcal{A},\epsilon}(0)=1]-\Pr[\mathsf{Expt\text{-}Comp\text{-}Smooth}_{\mathcal{A},\epsilon}(1)=1]\leq negl(\lambda),$$

where the experiment $\mathsf{Expt\text{-}Comp\text{-}Smooth}_{\mathcal{A},\epsilon}$ *is described in Figure 4.6.*

$\mathsf{Expt\text{-}Comp\text{-}Smooth}_{\mathcal{A},\epsilon}(\beta)$

The challenger samples RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$, 2 group elements $a, b \leftarrow \mathbb{Z}_N$ and $2n$ λ-bit primes $e_{i,c} \leftarrow \mathrm{PRIMES}(\lambda)$ for $i \in [n], c \in \{0,1\}$, and sets $K = (a, b, \{e_{i,c}\}_{i,c})$. It then samples a bit string $x \leftarrow \mathcal{X}$.

The challenger then sets j_ϵ to be the smallest index such that $p_{j_\epsilon} > (2^{n-m+2}\log N/\epsilon)^3$.

The challenger now computes (g, h) depending on challenge bit β in the following way.

- If $\beta = 0$,
  - Let the prime factorization of $\phi(N)$ be $\phi(N) = \prod r_i^{k_i}$, where $k_i \geq 1$, and r_i's are the distinct prime factors arranged in an increasing order. Let ℓ_N denotes number of distinct prime factors of $\phi(N)$.
  - It then sets $\widetilde{y} = (ax + b) \cdot \mathbf{e}_x \pmod{\phi(N)}$.
  - For each for $i \in [\ell_N]$ such that $r_i \leq p_{j_\epsilon}$, the challenger sets $y^{(i)} = \widetilde{y} \pmod{r_i^{k_i}}$. For each $i \in [\ell_N]$ such that $r_i > p_{j_\epsilon}$, it samples $y^{(i)} \leftarrow \mathbb{Z}_{r_i^{k_i}}$. The challenger then computes y which has CRT representation $(y^{(1)}, \ldots, y^{(\ell_N)})$.
  - It then samples a generator $g \leftarrow \mathbb{Z}_N^*$ and sets $h = g^{y \cdot f_1 \cdot f_2}$.
- If $\beta = 1$, the challenger samples generators $\tilde{g}, \tilde{h} \leftarrow \mathbb{Z}_N^*$ and sets $g = \tilde{g}^{\prod_{i=3}^{j_\epsilon} f_i}$, $h = \tilde{h}^{\prod_{i=1}^{j_\epsilon} f_i}$.

The challenger sends (N, g, K, h) to the adversary. The adversary then outputs a bit $\beta' \in \{0,1\}$, and the output of the experiment is set to be the same bit β'.

Figure 4.6: Security Game for Lemma 4.3.2

The proof of above lemma is identical to that of Lemma 4.2.2, and follows via a sequence of hybrids. Lastly, by combining Lemmas 4.3.1 and 4.3.2, Theorem 4.3.1 follows.

■

4.4 Φ-Hiding based Extractor Lemma

In this section, we prove a useful lemma that will help prove the security of our Φ-hiding-based constructions later. This has appeared (and implicitly used) in most existing Φ-hiding-based works. Here we abstract it out for ease of exposition.

Let $\mathsf{Ext} : \mathbb{Z}_N \times \mathbb{S} \to \mathcal{Y}$ be a $(\lambda - 1, \epsilon)$ strong extractor, where ϵ is negligible in the parameter λ. Informally, the lemmas states that, for every λ-bit prime e, applying extractor on an e^{th} root of a generator $g \in \mathbb{Z}_N^*$ is indistinguishable from random. Formally, we claim the following:

Lemma 4.4.1. *Assuming the Φ-hiding assumption holds, then for every admissible stateful PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda, \kappa \in \mathbb{N}$, such that $\kappa \geq 5\lambda$, the following hold,*

$$\Pr\left[\mathcal{A}(y_b) = b \;:\; \begin{array}{c} N \leftarrow \mathsf{RSA}(\kappa);\ \mathfrak{s} \leftarrow \mathbb{S} \\ e \leftarrow \mathrm{PRIMES}(\lambda);\ g \leftarrow \mathbb{Z}_N^* \\ F \leftarrow \mathcal{A}(N, \mathfrak{s}, e, g);\ b \leftarrow \{0,1\} \\ y_0 = \mathsf{Ext}(g^{F/e}, \mathfrak{s});\ y_1 \leftarrow \mathcal{Y} \end{array}\right] \leq negl(\lambda),$$

where $\mathcal{A}$ is an admissible adversary as long as $e \nmid F$.

Proof. The proof of this lemma follows a simple sequence of hybrids. First,

using Φ-hiding we can indistinguishably switch to sampling (e, N) such that $e \mid \phi(N)$. Once we have that $e \mid \phi(N)$, we know that there will exist e e^{th}-roots of generator g with all but a negligible probability. Thus, using the strong extractor guarantee, we get that y_0 is indistinguishable from random.

Claim 4.4.1. *Assuming the Φ-hiding assumption holds, then for every admissible stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for all $\lambda, \kappa \in \mathbb{N}$, such that $\kappa \geq 5\lambda$, the following holds*

$$\Pr\left[\mathcal{A}(y_b) = b \ : \ \begin{array}{c} N \leftarrow \mathsf{RSA}(\kappa);\ \mathfrak{s} \leftarrow \mathbb{S} \\ e \leftarrow \mathrm{PRIMES}(\lambda);\ g \leftarrow \mathbb{Z}_N^* \\ F \leftarrow \mathcal{A}(N, \mathfrak{s}, e, g);\ b \leftarrow \{0,1\} \\ y_0 = \mathsf{Ext}(g^{F/e}, \mathfrak{s});\ y_1 \leftarrow \mathcal{Y} \end{array}\right]$$
$$- \Pr\left[\mathcal{A}(y_b) = b \ : \ \begin{array}{c} e \leftarrow \mathrm{PRIMES}(\lambda);\ \mathfrak{s} \leftarrow \mathbb{S} \\ N \leftarrow \mathsf{RSA}_e(\kappa);\ h \leftarrow \mathbb{Z}_N^* \\ F \leftarrow \mathcal{A}(N, \mathfrak{s}, e, h^e);\ b \leftarrow \{0,1\} \\ y_0 = \mathsf{Ext}(h^F, \mathfrak{s});\ y_1 \leftarrow \mathcal{Y} \end{array}\right] \leq \mathit{negl}(\lambda),$$

where $\mathcal{A}$ is an admissible adversary as long as $e \nmid F$.

Proof. The proof of this lemma follows directly from the Φ-hiding assumption. Suppose there exists a PPT adversary $\mathcal{A}$ such that can distinguish between the two hybrid distributions with non-negligible probability γ. We use $\mathcal{A}$ to construct a reduction algorithm $\mathcal{B}$ that breaks the Φ-hiding assumption. Let e be a randomly chosen λ-bit prime. The Φ-hiding challenger samples an RSA modulus N and sends it to reduction algorithm $\mathcal{B}$. Given inputs N, e, the reduction algorithm samples a random seed $\mathfrak{s}$, and element $h \leftarrow \mathbb{Z}_N^*$. (Here it actually samples $h \leftarrow \mathbb{Z}_N$ and aborts whenever $h \notin \mathbb{Z}_N^*$. This happens with only negligible probability.) Next, it computes generator as $g = h^e$, and sends

parameters $(N, \mathfrak{s}, e, g)$ to $\mathcal{A}$. The adversary $\mathcal{A}$ sends an integer F to $\mathcal{B}$, and $\mathcal{B}$ first samples a random bit b, output $y_1 \leftarrow \mathcal{Y}$, and computes $y_0 = \mathsf{Ext}(h^F, \mathfrak{s})$. $\mathcal{B}$ sends y_b as the challenge to the adversary $\mathcal{A}$. If the adversary $\mathcal{A}$ outputs b, then $\mathcal{B}$ guesses that $e \nmid \phi(N)$, else it guesses that $e \mid \phi(N)$.

Let us now analyze the advantage of the reduction algorithm $\mathcal{B}$. Note that if $e \nmid \phi(N)$, then the distributions $\left\{(g, g^{F/e}) \ : \ g \leftarrow \mathbb{Z}_N^*\right\}$ and $\left\{(h^e, h^F) \ : \ h \leftarrow \mathbb{Z}_N^*\right\}$ are identically distributed as long as $e \nmid F$. Therefore, the reduction algorithm perfectly simulates the hybrid distributions for $\mathcal{A}$ depending on whether $e \mid \phi(N)$ or not. Hence, if $\mathcal{A}$ distinguishes with non-negligible probability γ, then $\mathcal{B}$'s advantage in Φ-hiding is also γ. Thus, the claim follows. ■

Claim 4.4.2. *If* Ext *is a* $(\lambda - 1, \epsilon)$ *strong extractor, where* ϵ *is negligible in the parameter* λ*, then for every admissible stateful adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for all* $\lambda, \kappa \in \mathbb{N}$*, such that* $\kappa \geq 5\lambda$*, the following holds*

$$\Pr\left[\mathcal{A}(y_b) = b \ : \ \begin{array}{c} e \leftarrow \mathrm{PRIMES}(\lambda);\ \mathfrak{s} \leftarrow \mathbb{S} \\ N \leftarrow \mathsf{RSA}_e(\kappa);\ h \leftarrow \mathbb{Z}_N^* \\ F \leftarrow \mathcal{A}(N, \mathfrak{s}, e, h^e);\ b \leftarrow \{0,1\} \\ y_0 = \mathsf{Ext}(h^F, \mathfrak{s});\ y_1 \leftarrow \mathcal{Y} \end{array}\right] \leq negl(\lambda),$$

where $\mathcal{A}$ *is an admissible adversary as long as* $e \nmid F$*.*

Proof. This follows directly from the strong extractor guarantee of Ext. The proof relies on the fact that for any given element $h \in \mathbb{Z}_N^*$, there exists $e - 1$ other distinct elements $\widetilde{h} \in \mathbb{Z}_N^*$, such that $h^e = \widetilde{h}^e$ when $e \mid \phi(N)$. Concretely,

for any $h \in \mathbb{Z}_N^*$, prime e, and integer F such that $e \nmid F$, let $S_{e,h}$ and $T_{e,h,F}$ denote the following sets:

$$S_{e,h} = \left\{ \widetilde{h} \in \mathbb{Z}_N^* \ : \ h^e = \widetilde{h}^e \right\}, \qquad T_{e,h,F} = \left\{ \widetilde{h} \in \mathbb{Z}_N^* \ : \ \widetilde{h} = g^F, g \in S_{e,h} \right\}.$$

Let $\mathcal{D}_{e,h,F}$ denote the uniform distribution over $T_{e,h,F}$. Note that $\mathcal{D}_{e,h,F}$ has min-entropy $\log_2 e > \lambda - 1$, since e is a λ-bit prime. Therefore, by extractor security the claim follows since ϵ is negligible. ■

From Claims 4.4.1 and 4.4.2, the lemma follows. ■

Chapter 5

Constructions

In this chapter, we finally describe our constructions. In Section 5.1, we present our one-way function with encryption construction from Φ-hiding assumption. In Section 5.2, we present our one-way function with encryption scheme based on q-DDHI assumption. In Section 5.3, we describe how to improve the efficiency of this construction with the use of pairings. Finally, in Sections 5.4 and 5.5, we describe our Hinting PRG constructions based on Φ-hiding and q-DDHI assumptions, respectively.

5.1 One-Way Function with Encryption from Φ-Hiding Assumption

In this section, we construct (k, n, ℓ)-recyclable one-way function with encryption (OWFE) from Φ-Hiding assumption. The construction assumes $k \geq 7\lambda$ and $n - k \leq \alpha \log n$ for any fixed constant α. For any parameters λ, ℓ, let $\mathsf{Ext}_{\lambda,\ell} : \{0,1\}^\lambda \times \mathcal{S} \to \{0,1\}^\ell$ be a $(\lambda - 1, \epsilon_{\mathsf{Ext}})$ strong seeded extractor, where ϵ_{Ext} is negligible in λ. [1] Let p_i denote the i^{th} (smallest) prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$ for all i. And, let f_i denote $p_i^{\widetilde{e_i}}$ for all i.

[1] Note that such an extractor exists for $\ell = c \cdot \lambda$ for some constant $c < 1$. The construction can be extended for any $\ell \geq \lambda$ with the help of PRGs.

The construction proceeds as follows.

$K(1^\lambda)$: On input security parameter λ and length ℓ, set RSA modulus length $\kappa = 5\lambda$, and sample RSA modulus $N \leftarrow \mathsf{RSA}(\kappa)$. Next, sample a generator $g \leftarrow \mathbb{Z}_N^*$, $2n$ (λ-bit) primes $e_{i,b} \leftarrow \mathrm{PRIMES}(\lambda)$ for $(i,b) \in [n] \times \{0,1\}$ and elements $d_0, d_1 \leftarrow \mathbb{Z}_N$. Then, sample a seed $\mathfrak{s} \leftarrow \mathcal{S}$ of extractor $\mathsf{Ext}_{\lambda,\ell}$ and output public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$.

$f(\mathsf{pp}, x)$: Let $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$. Output $y = g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_i e_{i,x_i}} \mod N$.

$E_1(\mathsf{pp}, (i,b); \rho)$: Parse pp as $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$. Output ciphertext $\mathsf{ct} = (g^{\rho \cdot e_{i,b}} \mod N, i, b)$.

$E_2(\mathsf{pp}, (y, i, b); \rho)$: Let pp be $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$. Compute $h = y^\rho \mod N$ and output $z = \mathsf{Ext}(h, \mathfrak{s})$.

$D(\mathsf{pp}, \mathsf{ct}, x)$: Let $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$. Parse ct as (t, i, b). If $b = x_i$, compute $h = t^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_{j \neq i} e_{j,x_j}} \mod N$ and output $\mathsf{Ext}(h, \mathfrak{s})$. Otherwise, output $\perp$.

Correctness. For any public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$, any string $x \in \{0,1\}^n$, any index $i \in [n]$, any randomness ρ, we have $D(\mathsf{pp}, E_1(\mathsf{pp}, (i, x_i); \rho), x) = g^{\rho f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_j e_{j,x_j}} = f(\mathsf{pp}, x)^\rho = E_2(\mathsf{pp}, (f(\mathsf{pp}, x), i, x_i); \rho)$.

5.1.1 Security

We now prove the one-wayness, encryption security, and smoothness properties of the above scheme.

One-Wayness. We now prove that the above construction satisifes (k, n)-one-wayness property when $k \geq 7\lambda$ and $n - k \leq \alpha \log n$ for any fixed constant α.

Theorem 5.1.1. *Assuming the Φ-hiding assumption holds, the above construction satisfies (k, n)-one-wayness property as per Definition 2.1.1.*

Proof. We first prove that no PPT adversary can win the following game with a non-negligible advantage assuming the Φ-hiding assumption. We then prove how a PPT adversary breaking the one-wayness property of the above scheme can be used to break the following game.

Game G: The challenger chooses RSA modulus $\kappa = 5\lambda$, samples $N \leftarrow \mathsf{RSA}(\kappa)$, prime $e \leftarrow \text{PRIMES}(\lambda)$ and a value $z \leftarrow \mathbb{Z}_N^*$. The challenger sends (N, e, z) to the adversary, which then outputs w. The adversary wins if $w^e = z \mod N$.

We now argue that no PPT adversary can win the above game with non-negligible probability.

Lemma 5.1.1. *Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the probability that $\mathcal{A}$ wins in Game G is at most negl(λ).*

Proof. We prove the lemma using the following intermediate Game H.

Game H: The challenger chooses RSA modulus $\kappa = 5\lambda$, samples prime $e \leftarrow \text{PRIMES}(\lambda)$ and $N \leftarrow \mathsf{RSA}(\kappa)$ s.t. $e|\phi(N)$. It then samples an element $z \leftarrow \mathbb{Z}_N^*$. The challenger sends (N, e, z) to the adversary, which then outputs w. The adversary wins if $w^e = z \mod N$.

Let the advantage of any adversary $\mathcal{A}$ in Game G be $\mathsf{Adv}_G^{\mathcal{A}}$ and in Game H be $\mathsf{Adv}_H^{\mathcal{A}}$.

Claim 5.1.1. *For every adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\mathsf{Adv}_H^{\mathcal{A}} \leq$ negl(λ).*

Proof. As $e|\phi(N)$, only a negligible fraction of $z \in \mathbb{Z}_N^*$ have a w s.t. $w^e = z \mod N$. Therefore, no PPT adversary can find a w s.t. $w^e = z \mod N$ with non-negligible probability. ∎

Claim 5.1.2. *Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $|\mathsf{Adv}_G^{\mathcal{A}} - \mathsf{Adv}_H^{\mathcal{A}}| \leq$ negl(λ).*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{Adv}_G^{\mathcal{A}} - \mathsf{Adv}_H^{\mathcal{A}}|$ is non-negligible. We construct a reduction algorithm that breaks Φ-hiding

assumption. $\mathcal{B}$ samples $e \leftarrow \text{PRIMES}(\lambda)$ and plays Φ-hiding game for e. The challenger sends RSA modulus N to $\mathcal{B}$, which samples $z \leftarrow \mathbb{Z}_N^*$ and sends (N, e, z) to $\mathcal{A}$. If $\mathcal{A}$ outputs w s.t. $w^e = z \mod N$, then $\mathcal{B}$ guesses that $\phi(N)$ is uniformly sampled from $\mathsf{RSA}(\kappa)$. Otherwise, it guesses that $e|\phi(N)$. ■

By the above 2 claims and triangle inequality, no PPT adversary can win Game G with a non-negligible advantage. ■

Lemma 5.1.2. *Assuming the Φ-hiding assumption holds, for every PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the advantage of $\mathcal{A}$ in (k, n)-one-wayness game is at most negl(λ).*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks (k, n)-one-wayness property of the encryption scheme with non-negligible probability ϵ. We construct a reduction algorithm $\mathcal{B}$ that wins against Game G challenger $\mathcal{C}$.

The adversary first sends a (k, n) source S to $\mathcal{B}$. The challenger $\mathcal{C}$ then sends (N, e, z) to $\mathcal{B}$. The reduction algorithm samples a bit string $x \leftarrow S$, an index $j \leftarrow [n]$, extractor seed $\mathfrak{s} \leftarrow \mathcal{S}$, exponents $d_0, d_1 \leftarrow \mathbb{Z}_p$ primes $e_{i,b'} \leftarrow \text{PRIMES}(\lambda)$ for $(i, b') \neq (j, 1 - x_j)$. It then sets generator $g = z$ and prime $e_{j,1-x_j} = e$. $\mathcal{B}$ then sends public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b'}\}_{i,b'}, d_0, d_1)$ and challenge $y = z^{\prod_i e_{i,x_i}} \mod N$ to the adversary. The adversary outputs x'. If $f(\mathsf{pp}, x') \neq f(\mathsf{pp}, x)$ or $x_j = x'_j$, then $\mathcal{B}$ aborts. Otherwise, we have $h^e = z^F \mod N$, where $F = f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_i e_{i,x_i}$ and $h = z^{f_1 \cdot f_2 \cdot (d_0 x' + d_1) \prod_{i \neq j} e_{i,x'_i}} \mod N$. As e is a randomly sampled λ-bit prime, $e \nmid F$ with overwhelming probability. $\mathcal{B}$ computes $z^{1/e} \mod N$ using Shamir's trick [93]. Concretely,

$\mathcal{B}$ first computes integers a, b s.t. $a \cdot e + b \cdot F = 1$ and outputs $w = h^b \cdot z^a \mod N$.

We now analyze the advantage of $\mathcal{B}$ in Game G. By our assumption, $f(\mathsf{pp}, x') = f(\mathsf{pp}, x)$ with non-negligible probability ϵ. We prove that $x' \neq x$ with non-negligible probability. As $k \geq \kappa + 2\lambda$, we know that for any pp, $\Pr_{x \leftarrow S}[\exists t \in \{0,1\}^n \text{ s.t. } x \neq t \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, t)] \geq 1 - \text{negl}(\lambda)$. Therefore, $\Pr[x' \neq x \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, x')] \geq \epsilon/2 - \text{negl}(\lambda)$ and $\Pr[x'_j \neq x_j \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, x')] \geq \epsilon/2n - \text{negl}(\lambda)$ as j is sampled uniformly from $[n]$. Note that if $\mathcal{B}$ does not abort, it outputs w s.t. $w^e = z \mod N$ with overwhelming probability. Therefore, $\mathcal{B}$ breaks Game G security with non-negligible probability $\epsilon/2n - \text{negl}(\lambda)$. ■

The proof of Theorem 5.1.1 follows from Lemma 5.1.2. ■

Security of Encryption. We now prove that the above construction satisifes encryption security property.

Theorem 5.1.2. *Assuming the Φ-hiding assumption holds, the above construction satisfies encryption security property as per Definition 2.1.2.*

Proof. We prove the above theorem via a sequence of following hybrids.

Hybrid H_0: This is same as original OWFE security of encryption game when the challenger chooses $\beta = 0$.

1. The adversary sends bit string $x \leftarrow \{0,1\}^n$ and index $j \in [n]$ to the challenger.
2. The challenger sets modulus length $\kappa = 5\lambda$ and samples $N \leftarrow \mathsf{RSA}(\kappa)$, generator $g \leftarrow \mathbb{Z}_N^*$, extractor seed $\mathfrak{s} \leftarrow \mathcal{S}$ and primes $e_{i,b} \leftarrow \mathrm{PRIMES}(\lambda)$ for $(i,b) \in [n] \times \{0,1\}$.
3. The challenger samples $\rho \leftarrow \mathbb{Z}_N$, computes $\mathsf{ct} = g^{\rho \cdot e_{j,1-x_j}}, z = \mathsf{Ext}(g^{\rho f_1 \cdot f_2 \cdot (d_0 x + d_1) \cdot \prod_i e_{i,x_i}} \mod N, \mathfrak{s})$.
4. The challenger sends $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}), \mathsf{ct}, z$ to the adversary $\mathcal{A}$, which outputs a bit α.

Hybrid H_1: This hybrid is similar to previous hybrid except for the following changes.

3. The challenger samples $\tilde{g} \leftarrow \mathbb{Z}_N^*$, computes $\mathsf{ct} = \tilde{g}$, $z = \mathsf{Ext}(\tilde{g}^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_i e_{i,x_i} \cdot e_{j,1-x_j}^{-1}} \mod N, \mathfrak{s})$.

Hybrid H_2: This hybrid is same as previous game except that the challenger samples z uniformly at random.

3. The challenger samples $\tilde{g} \leftarrow \mathbb{Z}_N^*$, computes $\mathsf{ct} = \tilde{g}, z \leftarrow \{0,1\}^\ell$.

Hybrid H_3: This is same as original OWFE security of encryption game when the challenger chooses $\beta = 1$.

3. The challenger samples $\rho \leftarrow \mathbb{Z}_N$, computes $\mathsf{ct} = g^{\rho \cdot e_{j,1-x_j}}, z \leftarrow \{0,1\}^\ell$.

For any PPT adversary $\mathcal{A}$, let the probability that $\mathcal{A}$ outputs 1 in Hybrid H_s be $p_s^{\mathcal{A}}$. We prove that Hybrids H_0 and H_3 are computationally indstinguishable via the sequence of following lemmas.

Lemma 5.1.3. *For any adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$, we have $|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}| \leq$ negl(λ).*

Proof. We first observe that for any N, prime $e \nmid \phi(N)$ and generator $g \in \mathbb{Z}_N^*$, the distribution of $g^{\rho \cdot e} \mod N$ for a randomly sampled $\rho \leftarrow \mathbb{Z}_{\phi(N)}$ is identical to the distribution $\tilde{g} \leftarrow \mathbb{Z}_N^*$. This follows from the fact that g and g^e are generators of $\mathbb{Z}_N^*$. For a randomly sampled λ bit prime e, we know that $e \nmid \phi(N)$ with overwhelming probability. Similarly, for a randomly sampled $\rho \leftarrow \mathbb{Z}_N$, we know that $\rho \in \mathbb{Z}_{\phi(N)}$ with overwhelming probability. As a result, $\{\tilde{g} : \tilde{g} \leftarrow \mathbb{Z}_N^*\}$ is statistically indistinguishable from $\{g^{\rho \cdot e} : g \leftarrow \mathbb{Z}_N^*, \rho \leftarrow \mathbb{Z}_N, e \leftarrow \text{PRIMES}(\lambda)\}$. By a similar argument, for any F, the distribution $\{(g^{\rho \cdot e} \mod N, g^{\rho \cdot F} \mod N) : g \leftarrow \mathbb{Z}_N^*, \rho \leftarrow \mathbb{Z}_N, e \leftarrow \text{PRIMES}(\lambda)\}$ is statistically indistinguishable from the distribution $\{(\tilde{g}, \tilde{g}^{F \cdot e^{-1}} \mod N) : \tilde{g} \leftarrow \mathbb{Z}_N^*, e \leftarrow \text{PRIMES}(\lambda)\}$. Therefore, for every adversary $\mathcal{A}$, $|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}| \leq \text{negl}(\lambda)$. ∎

Lemma 5.1.4. *Assuming the Φ-hiding assumption holds, for any PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$, we have $|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}| \leq$ negl(λ).*

Proof. The above lemma follows from ϕ-based Extractor lemma (Lemma 4.4.1).

Suppose there exists a PPT adversary $\mathcal{A}$ such that $|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that violates ϕ-based extractor lemma.

The extractor lemma challenger first samples $N \leftarrow \mathsf{RSA}(\kappa)$, $\mathfrak{s} \leftarrow \mathcal{S}$, $e \leftarrow \text{PRIMES}(\lambda)$, $\tilde{g} \leftarrow \mathbb{Z}_N^*$ and sends $(N, \mathfrak{s}, e, \tilde{g})$ to reduction algorithm $\mathcal{B}$. The adversary $\mathcal{A}$ then sends a string $x \in \{0,1\}^n$ and index $j \in [n]$ to $\mathcal{B}$. $\mathcal{B}$ samples generator g, values $d_0, d_1 \leftarrow \mathbb{Z}_N$, and primes $e_{i,b} \leftarrow \text{PRIMES}(\lambda)$ for $(i,b) \neq (j, 1-x_j)$. $\mathcal{B}$ then sets $e_{j,1-x_j} = e$ and computes $F = f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_i e_{i,x_i}$. If $e|F$, the reduction algorithm aborts and guesses randomly. As e is a λ-bit prime, this happens with negligible probability. If $e \nmid F$, then $\mathcal{B}$ sends F to the challenger, which samples a bit $\gamma \leftarrow \{0,1\}$. If $\gamma = 0$, $\mathcal{C}$ computes $\tilde{z} \leftarrow \mathsf{Ext}(\tilde{g}^{F/e}, \mathfrak{s})$. If $\gamma = 1$, $\mathcal{C}$ samples $\tilde{z} \leftarrow \{0,1\}^\ell$. The challenger sends $\tilde{z}$ to $\mathcal{B}$. The reduction algorithm sets $\mathsf{ct} = \tilde{g}, z = \tilde{z}$ and sends $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1), \mathsf{ct}, z$ to $\mathcal{A}$. The adversary outputs a bit α. $\mathcal{B}$ outputs α as its guess in extractor lemma game.

Note that if $\gamma = 0$, then the distribution of $\mathsf{pp}, \mathsf{ct}, z$ sent by $\mathcal{B}$ is statistically indistinguishable from that of Hybrid H_1 challenger. If $\gamma = 1$, then the distribution of $\mathsf{pp}, \mathsf{ct}, z$ sent by $\mathcal{B}$ is statistically indistinguishable from that of H_2 challenger. Consequently if $\mathcal{B}$ does not abort, the advantage $|\Pr[\alpha = 1|\gamma - 0] - \Pr[\alpha = 1|\gamma = 1]| \geq |p_1^{\mathcal{A}} - p_2^{\mathcal{A}}| - \text{negl}(\lambda)$ is non-negligible. As $\mathcal{B}$ aborts with only negligible probability, it wins the extractor lemma game with non-negligible probability. ■

Lemma 5.1.5. *For any PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$, we have $|p_2^{\mathcal{A}} - p_3^{\mathcal{A}}| \leq$*

$negl(\lambda)$.

Proof. The distribution $\{\tilde{g} : \tilde{g} \leftarrow \mathbb{Z}_N^*\}$ is statistically indistinguishable from $\{g^{\rho \cdot e} : g \leftarrow \mathbb{Z}_N^*, \rho \leftarrow \mathbb{Z}_N, e \leftarrow \text{PRIMES}(\lambda)\}$ as mentioned in proof of Claim 5.1.3. ■

By the above lemmas and triangle theorem, no PPT adversary can distinguish between Hybrids H_0 and H_3 with non-negligible probability assuming the Φ-hiding assumption. ■

Smoothness. We now prove that the above construction satisifes (k, n)-smoothness property when $k \geq 7\lambda$ and $n - k \leq \alpha \log n$ for any fixed constant α.

Theorem 5.1.3. *Assuming the Φ-hiding assumption holds, the above construction satisfies (k, n)-smoothness security property as per Definition 2.1.3.*

Proof. First, we introduce a useful notation. For any constant $\epsilon > 0$, let j_ϵ be the smallest index such that $p_{j_\epsilon} > (2^{n-k+2} \log N/\epsilon)^3$. Note that $(2^{n-k+2} \log N/\epsilon)^3$ is polynomial in λ for the given setting of parameters. The proof of security follows via a sequence of hybrids. Below we first describe the sequence of hybrids and later argue indistinguishability to complete the proof. At a very high level, the proof structure is somewhat similar to that used in [104], where for proving security, one first assumes (for the sake of contradiction) that the adversary wins with some non-negligible probability δ and then, depending upon δ, one could describe a sequence of hybrids such that no PPT adversary

can win with probability more than $2\delta/3$. This acts as a contradiction, thereby completing the proof.

For any PPT adversary $\mathcal{A}$, let $p_s^{\mathcal{A}}$ be the probability that $\mathcal{A}$ outputs 1 in Hybrid H_s. For the sake of contradiction, we assume that $\mathcal{A}$ breaks (k, n)-smoothness property with non-negligible advantage $\delta(\lambda)$ i.e., there exists a polynomial $v(\cdot)$ s.t. $|p_0^{\mathcal{A}} - p_2^{\mathcal{A}}| = \delta(\lambda) > \frac{1}{v(\lambda)}$ for infinitely often $\lambda \in \mathbb{N}$. Let $\epsilon = \frac{1}{2v(\lambda)}$. We provide a non-uniform reduction where the description of hybrids and the reduction algorithm depends on ϵ.

Hybrid H_0: This is the same as the original smoothness security game, except that the challenger always chooses source S_0.

1. The adversary first sends two (k, n) sources S_0, S_1 to the challenger. The challenger sets modulus length $\kappa = 5\lambda$ and samples $N \leftarrow \mathsf{RSA}(\kappa)$, extractor seed $\mathfrak{s} \leftarrow \mathcal{S}$, elements $d_0, d_1 \leftarrow \mathbb{Z}_N$ and primes $e_{i,b} \leftarrow \text{PRIMES}(\lambda)$ for $(i, b) \in [n] \times \{0, 1\}$.
2. The challenger then samples a generator $g \leftarrow \mathbb{Z}_N^*$ and sets public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$.
3. The challenger samples $x \leftarrow S_0$ and sends $\mathsf{pp}, y = g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_{i=1}^{n} e_{i,x_i}} \mod N$ to the adversary.
4. The adversary outputs a bit b'.

Hybrid H_1: In this hybrid, the challenger does not sample x and picks the challenge y from a uniform distribution.

2. The challenger then samples a generator $\tilde{g} \leftarrow \mathbb{Z}_N^*$, sets $g = \tilde{g}^{\prod_{i=3}^{j_\epsilon} f_i}$ and sets public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$.

3. The challenger samples $z \leftarrow \mathbb{Z}_N^*$ and sends $\mathsf{pp}, y = z^{\prod_{i=1}^{j_\epsilon} f_i} \mod N$ to the adversary.

Hybrid H_2: This is same as the original smoothness security game, except that the challenger always chooses source S_1.

2. The challenger then samples a generator $g \leftarrow \mathbb{Z}_N^*$ and sets public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$.

3. The challenger samples $x \leftarrow S_1$ and sends $\mathsf{pp}, y = g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_{i=1}^{n} e_{i,x_i}} \mod N$ to the adversary.

Lemma 5.1.6. *Assuming the Φ-hiding assumption holds, for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\delta(\lambda) \geq 2\epsilon = 1/v(\lambda)$, we have $|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}| \leq \epsilon/2 + negl(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that has a non-negligible advantage $\delta(\lambda)$ in smoothness game, and can distinguish between Hybrids H_0 and H_1 with probability $\epsilon/2 + \gamma$ for some non-negligible value γ. We construct a reduction algorithm $\mathcal{B}$ that breaks our strengthened hashing lemma (Theorem 4.3.1) and thereby breaking the Φ-hiding assumption.

The adversary $\mathcal{A}$ sends two (k, n)-sources S_0, S_1 to the reduction algoritm $\mathcal{B}$. $\mathcal{B}$ plays hashing lemma game for source S_0 with the challenger $\mathcal{C}$. The

hashing lemma challenger $\mathcal{C}$ sends $(N, g, a, b, \{e_{i,b}\}_{i,b}, y)$ to the reduction algorithm $\mathcal{B}$. The reduction algorithm samples a seed $\mathfrak{s} \leftarrow S$, sets $d_0 = a, d_1 = b$ and sends public parameters $\mathsf{pp} = (N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1)$, challenge y to the adverary $\mathcal{A}$. The adversary outputs a bit b'. $\mathcal{B}$ outputs b' as its guess in hashing lemma game.

Let us analyze advantage of $\mathcal{B}$ in hashing lemma game. If the challenger samples $g \leftarrow \mathbb{Z}_N^*, x \leftarrow S_0, y = g^{f_1 \cdot f_2 \cdot (ax+b) \prod_{i=1}^{n} e_{i,x_i}} \mod N$, then $\mathcal{B}$ emulates Hybrid H_0 challenger to $\mathcal{A}$. If the challenger samples $\tilde{g} \leftarrow \mathbb{Z}_N^*, z \leftarrow \mathbb{Z}_N^*$ and sets $g = \tilde{g}^{\prod_{i=3}^{j_\epsilon} f_i}$, $y = z^{\prod_{i=1}^{j_\epsilon} f_i}$, then $\mathcal{B}$ emulates Hybrid H_1 challenger to $\mathcal{A}$. Therefore, $\mathcal{B}$ breaks hashing lemma game with advantage $|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}| \geq \epsilon/2 + \gamma$. ■

Lemma 5.1.7. *Assuming the Φ-hiding assumption holds, for any PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\delta(\lambda) \geq 2\epsilon = 1/v(\lambda)$, we have $|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}| \leq \epsilon/2 + negl(\lambda)$.*

Proof. This proof is similar to the proof of Lemma 5.1.6. ■

By the above 2 lemmas and triangle inequality, $\mathcal{A}$ can distinguish between Hybrids H_0 and H_2 with probability at most $\epsilon + \text{negl}(\lambda) < 2\delta/3$. This contradicts the assumption that $\mathcal{A}$ has an advantage of δ.[2] Therefore, no PPT adversary can break the (k, n)-smoothness property of the above construction with non-negligible probability. ■

[2]Note that the contradiction does not happen when δ is negligible. If δ is negligible, then j_ϵ is superpolynomial, and the reduction algorithm takes superpolynomial time to execute.

5.2 One Way Function with Encryption from q-DDHI Assumption

In this section, we construct (k, n)-OWFE from any n-DDHI hard group generator GGen. Suppose $\mathsf{GGen}(1^\lambda)$ generates a group of order at most 2^m, the below construction requires $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$ for any fixed constant α. For the sake of simplicity, we construct an OWFE scheme where the encryption algorithm outputs elements in a group. The construction can be extended to output ℓ-length bit strings using PRGs and randomness extractors. We present a variant of this construction with shorter ciphertext from n-DBDHI assumption (using pairings) in Section 5.3.

$K(1^\lambda)$: Sample a group $\mathcal{G} = (\mathbb{G}, p) \leftarrow \mathsf{GGen}(1^\lambda)$. Sample a generator $g \leftarrow \mathbb{G}_1$ and random values $\alpha, d_0, d_1 \leftarrow \mathbb{Z}_p$. Output the public parameters $(\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.

$f(\mathsf{pp}, x)$: Parse public parameters pp as $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$. Let the polynomial $(d_0 x + d_1) \cdot \prod_{j=1}^{n} (\alpha + 2j + x_j) = \sum_{i=0}^{n} c_i \alpha^i$, where c_i is a function of d_0, d_1, x. Output $\prod_{i=0}^{n} \left(g^{\alpha^i}\right)^{c_i}$.

$E_1(\mathsf{pp}, (i, b); \rho)$: Let $h = g^{\rho \cdot (\alpha + 2i + b)}$. Compute and output $(h, h^\alpha, h^{\alpha^2}, \cdots, h^{\alpha^{n-1}}, i)$. Note that these values can be computed given $(g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n})$.

$E_2(\mathsf{pp}, (y, i, b); \rho)$: Compute and output y^ρ.

$D(\mathsf{pp}, \mathsf{ct}, x)$: Let $\mathsf{ct} = (h, h^\alpha, h^{\alpha^2}, \cdots, h^{\alpha^{n-1}}, i)$. Consider the polynomial

$(d_0 x + d_1) \cdot \prod_{j \in [1,n] \setminus \{i\}} (\alpha + 2j + x_j) = \sum_{j=0}^{n-1} c_j \alpha^j$, where c_j is a function of d_0, d_1, x. Compute and output $\prod_{j=0}^{n-1} \left(h^{\alpha^j}\right)^{c_j}$.

Correctness. For any set of public parameters $\mathsf{pp} = (\mathcal{G}, g, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$, string $x \in \{0,1\}^n$, index $j \in [n]$ and randomness ρ, we have $D(\mathsf{pp}, E_1(\mathsf{pp}, (j, x_j); \rho), x) = g^{\rho(d_0 x + d_1) \prod_i (\alpha + 2i + x_i)} = f(\mathsf{pp}, x)^{\rho} = E_2(\mathsf{pp}, (f(\mathsf{pp}, x), j, x_j); \rho)$.

5.2.1 Security

We now prove that the above construction satisfies one-wayness, encryption security, and smoothness properties.

One-Wayness. We now prove that the above construction satisfies (k, n)-one-wayness property for any $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$.

Lemma 5.2.1. *Assuming n-DDHI assumption holds (Assumption 3), for any (k, n) source s.t. $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$, the above construction satisfies (k, n)-one-wayness property as per Definition 2.1.1.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the one-wayness property of the above construction with non-negligible probability. We construct a reduction algorithm $\mathcal{B}$ that wins n-DDHI game with non-negligible probability.

The adversary $\mathcal{A}$ first sends a (k, n)-source S to the reduction algorithm $\mathcal{B}$. The challenger then sends $(\mathcal{G}, h, h^{\alpha}, h^{\alpha^2}, \cdots, h^{\alpha^n}, T)$ to the reduction algorithm $\mathcal{B}$. The reduction algorithm samples a string $x \leftarrow S$,

$d_0, d_1 \leftarrow \mathbb{Z}_p$, computes public parameters $\mathsf{pp} = (\mathcal{G}, h, h^{\alpha}, h^{\alpha^2}, \cdots h^{\alpha^n}, d_0, d_1)$ and sends $\mathsf{pp}, y = f(\mathsf{pp}, x)$ to the adversary $\mathcal{A}$. The adversary outputs a string x'. If $x' = x$ or $f(\mathsf{pp}, x) \neq f(\mathsf{pp}, x')$, the reduction algorithm aborts and outputs a random bit. Otherwise, $\mathcal{B}$ computes α s.t. $(d_0 x + d_1) \cdot \prod_{i=1}^{n}(\alpha + 2i + x_i) = (d_0 x' + d_1) \cdot \prod_{i=1}^{n}(\alpha + 2i + x'_i) \mod p$. The reduction algorithm then checks if $T = g^{1/\alpha}$. If $T = g^{1/\alpha}$, it outpus 1. Otherwise, it outputs 0.

We now analyze the advantage of $\mathcal{B}$ in n-DDHI game. By our assumption, $f(\mathsf{pp}, x') = f(\mathsf{pp}, x)$ with non-negligible probability ϵ. We prove that the reduction algorithm does not abort with non-negligible probability. As $k \geq m + 2\lambda$, we know that for any pp, $\Pr_{x \leftarrow S}[\exists t \in \{0,1\}^n \text{ s.t. } x \neq t \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, t)] \geq 1 - \mathrm{negl}(\lambda)$. Therefore, $\Pr[x' \neq x \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, x')] \geq \epsilon/2 - \mathrm{negl}(\lambda)$. Note that if $\mathcal{B}$ does not abort, it breaks the n-DDHI game with advantage $1/2$. Therefore, the overall advantage of $\mathcal{B}$ in breaking n-DDHI game is $\epsilon/4 - \mathrm{negl}(\lambda)$. ■

Security of Encryption. We now prove that the above construction satisfies encryption security property.

Lemma 5.2.2. *Assuming n-DDHI assumption holds (Assumption 3), the above construction satisfies encryption security property as per Definition 2.1.2.*

Proof. This is similar to the proof of Lemma 5.3.3. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks encryption security of the above construction with non-negligible probability. We construct a reduction algorithm $\mathcal{B}$ that wins against n-DDHI challenger $\mathcal{C}$.

The challenger $\mathcal{C}$ first samples a group structure $\mathcal{G} = (\mathbb{G}, p) \leftarrow \mathsf{GGen}(1^\lambda)$, a generator $h \leftarrow \mathbb{G}$, a value $\beta \leftarrow \mathbb{Z}_p^*$ and a bit $\gamma \leftarrow \{0,1\}$. If $\gamma = 0$, it sets $T = h^{1/\beta}$. Otherwise, it samples $T \leftarrow \mathbb{G}$. The challenger then sends $(\mathcal{G}, h, h^\beta, h^{\beta^2}, \cdots, h^{\beta^n}, T)$ to the reduction algorithm $\mathcal{B}$. The adversary sends a string $x \in \{0,1\}^n$ and an index j to $\mathcal{B}$. $\mathcal{B}$ samples $d_0, d_1 \leftarrow \mathbb{Z}_p$ and implicitly sets $\alpha = \beta - 2j - 1 + x_j$. It then computes public parameters $\mathsf{pp} = (\mathcal{G}, h, h^\alpha, h^{\alpha^2}, \cdots, h^{\alpha^n}, d_0, d_1)$, samples randomness $\rho \leftarrow \mathbb{Z}_p$ and computes $\mathsf{ct}^* = (h^\rho, h^{\rho\cdot\alpha}, h^{\rho\cdot\alpha^2}, \cdots, h^{\rho\cdot\alpha^{n-1}}, j)$. Consider the polynomial

$$\frac{\rho \cdot (d_0 x + d_1) \cdot \prod_{i=1}^{n}(\alpha + 2i + x_i)}{\alpha + 2j + 1 - x_j} = \frac{c}{\beta} + \sum_{i=0}^{n-1} c_i \beta^i$$

where $c, \{c_i\}_i$ are dependent only on ρ, x, d_0, d_1. The reduction algorithm computes $k^* = T^c \cdot \prod_{i=0}^{n-1} \left(h^{\beta^i}\right)^{c_i}$ and sends $\mathsf{pp}, \mathsf{ct}^*, k^*$ to the adversary. The adversary outputs a bit γ'. $\mathcal{B}$ outputs γ' as its guess in n-DDHI game.

We now analyze the advantage of $\mathcal{B}$ in n-DDHI game. As β is sampled uniformly, α is also uniformly distributed. Let $\rho' = \frac{\rho}{\alpha+2j+1-x_j} \mod p$. As $\beta \neq 0 \mod p$ and ρ is uniformly distributed, ρ' is also uniformly distributed in $\mathbb{Z}_p$. If $\gamma = 0$, then $(\mathsf{pp}, \mathsf{ct}^*, k^*)$ is same as $(\mathsf{pp}, E_1(\mathsf{pp}, (j, 1 - x_j); \rho'), E_2(\mathsf{pp}, (f(\mathsf{pp}, x), j, 1 - x_j); \rho')$. If $\gamma = 1$, then k^* is uniformly random. As $\mathcal{A}$ distinguishes these 2 distributions with non-negligible probability, $|\Pr[\gamma' = 1|\gamma = 0] - \Pr[\gamma' = 1|\gamma = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks n-DDHI assumption. ■

Smoothness. We now prove that the above construction satisfies (k, n)-smoothness property for any $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$.

Lemma 5.2.3. *The above construction satisfies (k, n)-smoothness property for any $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$ as per Definition 2.1.3.*

Proof. This proof is the same as the proof of Lemma 5.3.4. ■

5.3 One-Way Function with Encryption from q-DBDHI Assumption

We now construct (k, n, ℓ)-OWFE from any n-DBDHI hard group generator GGen. Suppose $\mathsf{GGen}(1^\lambda)$ generates a group of size $\theta(2^m)$, the below construction requires $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$. For the sake of simplicity, we construct an OWFE scheme where the encryption algorithm outputs elements in a group. The construction can be extended to output ℓ-length bit strings using PRGs and randomness extractors. This is a variant of the construction from the n-DDHI assumption presented in Section 5.2. This construction has the advantage of having a shorter ciphertext size.

$K(1^\lambda)$: Sample a group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_T, e, p) \leftarrow \mathsf{GGen}(1^\lambda)$. Sample a generator $g \leftarrow \mathbb{G}_1$ and random values $\alpha, d_0, d_1 \leftarrow \mathbb{Z}_p$. Output the public parameters $(\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.

$f(\mathsf{pp}, x)$: Parse public parameters pp as $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$. Let the polynomial $(d_0 x + d_1) \cdot \prod_{j=1}^{n} (\alpha + 2j + x_j) = \sum_{i=0}^{n} c_i \alpha^i$, where c_i is a function of d_0, d_1, x. Output $\prod_{i=0}^{n} \left(g^{\alpha^i}\right)^{c_i}$.

$E_1(\mathsf{pp}, (i, b); r)$: Compute and output $(g^{r \cdot (\alpha + 2i + b)}, i)$.

$E_2(\mathsf{pp}, (y, i, b); r)$: Compute and output $e(g^r, y)$.

$D(\mathsf{pp}, \mathsf{ct}, x)$: Let $\mathsf{ct} = (\mathsf{ct}', i)$. Consider the polynomial $(d_0 x + d_1) \cdot \prod_{j \neq i}(\alpha + 2j + x_j) = \sum_{j=0}^{n-1} c_j \alpha^j$, where c_j is a function of d_0, d_1, x. Compute and output $e\Big(\mathsf{ct}', \prod_{j=0}^{n-1} \Big(g^{\alpha^j}\Big)^{c_j}\Big)$.

Correctness. For any set of public parameters $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$, string $x \in \{0,1\}^n$, index $j \in [n]$ and randomness r, we have $\mathsf{ct} = E_1(\mathsf{pp}, (j, x_j); r) = (g^{r \cdot (\alpha + 2j + x_j)}, j)$ and $D(\mathsf{pp}, \mathsf{ct}, x) = e(g,g)^{r \cdot (d_0 x + d_1) \prod_i (\alpha + 2i + x_i)} = e(g^r, f(\mathsf{pp}, x)) = E_2(\mathsf{pp}, (f(\mathsf{pp}, x), j, x_j); r)$.

5.3.1 Security

We now prove that the above construction satisfies one-wayness, encryption security, and smoothness properties. Before that, we first prove a lemma which would be helpful in the security proofs.

Lemma 5.3.1. *For any prime p, any (k,n)-source S s.t. $k \geq \log p + 2\lambda$, $n \leq k + \log p - 2\lambda$, any subset $\mathcal{A} \subseteq [0, p-2n-2]$, the distribution $\Big(K, H(K,S)\Big)$ is statistically indistinguishable from $\Big(K, U\Big)$, where $H : \mathcal{K} \times \{0,1\}^n \to \mathbb{Z}_p$ is a hash function with key space $\mathcal{K} = (\mathbb{Z}_p \times \mathbb{Z}_p \times \mathcal{A})$ and is defined as $H((d_0, d_1, \alpha), x) = (d_0 x + d_1) \cdot \prod_{k=1}^{n} (\alpha + 2k + x_k) \mod p$, hash key $K = (d_0, d_1, \alpha)$ is uniformly sampled from $\mathcal{K}$ and U is the uniform distribution on $\mathbb{Z}_p$.*

Proof. This proof is information-theoretic. We first bound the probability $\Pr_K[H\ (K, s) = H(K, t)]$ for any $s, t \in \{0,1\}^n$ s.t. $s \neq t$. We then use analysis

similar to leftover hash lemma and prove that the distribution $(K, H(K,S))$ is statistically indistinguishable from (K,U). Consider any $s,t \in \{0,1\}^n$ s.t. $s \neq t$.

$$\Pr_K[H(K,s) = H(K,t)] = \sum_{c \in \mathbb{Z}_p} \Pr_K[H(K,s) = H(K,t) = c]$$

$$= \sum_{c \in \mathbb{Z}_p} \Pr_{d_0,d_1,\alpha} \Big[(d_0 s + d_1) = c \cdot \prod_{k=1}^{n} (\alpha + 2k + s_k)^{-1} \mod p \bigwedge$$

$$(d_0 t + d_1) = c \cdot \prod_{k=1}^{n} (\alpha + 2k + t_k)^{-1} \mod p \Big]$$

Note that for any $\alpha \in \mathcal{A}$, $(\alpha + 2k + s_k)^{-1}, (\alpha + 2k + t_k)^{-1} \mod p$ is unique for all $k \in [n]$. For any $\alpha \in \mathcal{A}$ and $c \in \mathbb{Z}_p$, let us compute the number of (d_0, d_1) pairs in $\mathbb{Z}_p^2$ that satisfy the above pair of equations. On subtracting the equations, we get

$$d_0(s-t) = c \cdot \Big(\prod_{k=1}^{n} (\alpha + 2k + s_k)^{-1} - \prod_{k=1}^{n} (\alpha + 2k + t_k)^{-1} \Big) \mod p.$$

Consider the following 2 cases.

- Case 1 ($s - t \neq 0 \mod p$): There exists a unique (d_0, d_1) pair satisfying the pair of equations for every $\alpha \in \mathcal{A}$ and $c \in \mathbb{Z}_p$. Therefore,

$$\sum_{c \in \mathbb{Z}_p} \Pr_K[H(K,s) = H(K,t) = c] = \sum_{c \in \mathbb{Z}_p} \frac{1}{p^2} = \frac{1}{p}$$

- Case 2 ($s - t = 0 \mod p$): If $c = 0$, for any $\alpha \in \mathcal{A}$, number of $(d_0, d_1) \in \mathbb{Z}_p^2$ satisfying the pair of equations is p. If $c \neq 0$, for any $\alpha \in \mathcal{A}$ such that $f(\alpha) = \prod_{k=1}^{n} (\alpha + 2k + s_k) - \prod_{k=1}^{n} (\alpha + 2k + t_k) = 0 \mod p$,

number of $(d_0, d_1) \in \mathbb{Z}_p^2$ satisfying the pair of equations is p. For any α s.t. $f(\alpha) \neq 0 \mod p$, no $(d_0, d_1) \in \mathbb{Z}_p^2$ satisfying the pair of equations. By lagrange's theorem, $f(\alpha) = 0 \mod p$ has at most n solutions.

$$\begin{aligned}
&\sum_{c \in \mathbb{Z}_p} \Pr_K[H(K,s) = H(K,t) = c] \\
&\leq \Pr_K[H(K,s) = H(K,t) = 0] + \sum_{c \neq 0} \Pr_K[H(K,s) = H(K,t) = c] \\
&\leq \frac{p}{p^2} + \sum_{c \neq 0} \frac{p}{p^2} \cdot \Pr_\alpha \Big[\prod_{k=1}^{n} (\alpha + 2k + s_k) - \prod_{k=1}^{n} (\alpha + 2k + t_k) = 0 \mod p \Big] \\
&\leq \frac{1}{p} + (p-1) \cdot \frac{1}{p} \cdot \frac{n}{p - 2n - 1} \leq \frac{2n+1}{p} \qquad \text{(Assuming } p - 2n - 1 \geq p/2\text{)}
\end{aligned}$$

We now bound the statistical distance between distributions $\mathcal{D}_1 = (K, H(K,S))$ and $\mathcal{D}_2 = (K, U)$. For any distribution D, let $\mathsf{CP}(D)$ be collision probability on D.

$$\begin{aligned}
\mathsf{CP}(D_1) &= \Pr_{\substack{K_1, K_2, \\ s,t \leftarrow S}} [(K_1, H(K_1,s)) = (K_2, H(K_2,t))] \\
&= \Pr[K_1 = K_2] \cdot \Pr_{\substack{K, \\ s,t \leftarrow S}} [H(K,s) = H(K,t)] \\
&= \frac{1}{|\mathcal{K}|} \cdot \Big(\Pr_{s,t}[s = t] + \Pr_{s,t}[s \neq t \mod p] \Pr_H[H(s) = H(t) | s \neq t \mod p] \\
&\qquad + \Pr_{s,t}[s = t \mod p, s \neq t] \Pr_H[H(s) = H(t) | s = t \mod p, s \neq t] \Big) \\
&\leq \frac{1}{|\mathcal{K}|} \cdot \Big(\frac{1}{2^k} + 1 \cdot \frac{1}{p} + \frac{1}{2^k} \cdot \Big\lfloor \frac{2^n}{p} \Big\rfloor \cdot \frac{2n+1}{p} \Big) \\
&\leq \frac{1}{|\mathcal{K}|} \cdot \Big(\frac{1}{2^k} + \frac{1}{p} + \frac{2^{n-k} \cdot (2n+1)}{p^2} \Big)
\end{aligned}$$

We know that statistical difference between D_1 and D_2 is given by

$$\begin{aligned}
\mathsf{SD}(D_1, D_2) &\leq \sqrt{|\mathcal{K}| \cdot p}\sqrt{\mathsf{CP}(D_1) - \mathsf{CP}(D_2)} \\
&\leq \sqrt{|\mathcal{K}| \cdot p}\sqrt{\frac{1}{|\mathcal{K}|}\Big(\frac{1}{2^k} + \frac{1}{p} + \frac{2^{n-k} \cdot (2n+1)}{p^2}\Big) - \frac{1}{|\mathcal{K}| \cdot p}} \\
&= \sqrt{\frac{p}{2^k} + \frac{2^{n-k} \cdot (2n+1)}{p}} \\
&= \mathrm{negl}(\lambda) \qquad (\text{As } k \geq \log p + 2\lambda \text{ and } n - k \leq \log p - 2\lambda)
\end{aligned}$$

■

One-Wayness. We now prove that the above construction satisfies (k, n)-one-wayness property for any $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$.

Lemma 5.3.2. *Assuming n-DBDHI assumption holds (Assumption 4), the above construction satisfies (k, n)-one-wayness property for any (k, n) s.t. $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$ as per Definition 2.1.1.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the one-wayness property of the above construction with non-negligible probability. We construct a reduction algorithm $\mathcal{B}$ that wins n-DBDHI game with non-negligible probability.

The adversary $\mathcal{A}$ first sends a (k, n)-source S to the reduction algorithm $\mathcal{B}$. The challenger then sends $(\mathcal{G}, h, h^{\alpha}, h^{\alpha^2}, \cdots, h^{\alpha^n}, T)$ to the reduction algorithm $\mathcal{B}$. The reduction algorithm samples a string $x \leftarrow S$, $d_0, d_1 \leftarrow \mathbb{Z}_p$, computes public parameters $\mathsf{pp} = (\mathcal{G}, h, h^{\alpha}, h^{\alpha^2}, \cdots h^{\alpha^n}, d_0, d_1)$

and sends $\mathsf{pp}, y = f(\mathsf{pp}, x)$ to the adversary $\mathcal{A}$. The adversary outputs a string x'. If $x' = x$ or $f(\mathsf{pp}, x) \neq f(\mathsf{pp}, x')$, the reduction algorithm aborts and outputs a random bit. Otherwise, $\mathcal{B}$ computes α s.t. $(d_0 x + d_1) \cdot \prod_{i=1}^{n}(\alpha + 2i + x_i) = (d_0 x' + d_1) \cdot \prod_{i=1}^{n}(\alpha + 2i + x'_i) \mod p$. The reduction algorithm then checks if $T = e(g,g)^{1/\alpha}$. If $T = e(g,g)^{1/\alpha}$, it outpus 1. Otherwise, it outputs 0.

We now analyze the advantage of $\mathcal{B}$ in n-DBDHI game. By our assumption, $f(\mathsf{pp}, x') = f(\mathsf{pp}, x)$ with non-negligible probability ϵ. We prove that the reduction algorithm does not abort with non-negligible probability. As $k \geq m + 2\lambda$, we know that for any pp, $\Pr_{x \leftarrow S}[\exists t \in \{0,1\}^n \text{ s.t. } x \neq t \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, t)] \geq 1 - \mathrm{negl}(\lambda)$. Therefore, $\Pr[x' \neq x \wedge f(\mathsf{pp}, x) = f(\mathsf{pp}, x')] \geq \epsilon/2 - \mathrm{negl}(\lambda)$. Note that if $\mathcal{B}$ does not abort, it breaks the n-DBDHI game with advantage $1/2$. Therefore, the overall advantage of $\mathcal{B}$ in breaking n-DBDHI game is $\epsilon/4 - \mathrm{negl}(\lambda)$. ■

Security of Encryption. We now prove that the above construction satisfies encryption security property.

Lemma 5.3.3. *Assuming n-DBDHI assumption holds (Assumption 4), the above construction satisfies encryption security property as per Definition 2.1.2.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks encryption security of the above construction with non-negligible probability. We construct a reduction algorithm $\mathcal{B}$ that wins n-DBDHI game with non-negligible probability.

The challenger $\mathcal{C}$ first samples a group structure $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_T, e, p) \leftarrow \mathsf{GGen}(1^\lambda)$, a generator $h \leftarrow \mathbb{G}_1$, a value $\beta \leftarrow \mathbb{Z}_p^*$ and a bit $\gamma \leftarrow \{0,1\}$. If $\gamma = 0$, it sets $T = e(h,h)^{1/\beta}$. Otherwise, it samples $T \leftarrow \mathbb{G}_T$. The challenger then sends $(\mathcal{G}, h, h^\beta, h^{\beta^2}, \cdots, h^{\beta^n}, T)$ to the reduction algorithm $\mathcal{B}$. The adversary sends a string $x \in \{0,1\}^n$ and an index j to $\mathcal{B}$. $\mathcal{B}$ samples $d_0, d_1 \leftarrow \mathbb{Z}_p$ and implicitly sets $\alpha = \beta - 2j - 1 + x_j$. It then computes public parameters $\mathsf{pp} = (\mathcal{G}, h, h^\alpha, h^{\alpha^2}, \cdots, h^{\alpha^n}, d_0, d_1)$, samples $\rho \leftarrow \mathbb{Z}_p$ and implicitly uses $h^{\rho/(\alpha+2j+1-x_j)}$ as randomness for encryption. It computes $\mathsf{ct}^* = (h^\rho, j)$. Consider the polynomial

$$\frac{\rho \cdot (d_0 x + d_1) \cdot \prod_{i=1}^{n} (\alpha + 2i + x_i)}{\alpha + 2j + 1 - x_j} = \frac{c}{\beta} + \sum_{i=0}^{n-1} c_i \beta^i$$

where $c, \{c_i\}_i$ are dependent only on ρ, x, d_0, d_1. The reduction algorithm computes $k^* = T^c \cdot e\Big(h, \prod_{i=0}^{n-1} \left(h^{\beta^i}\right)^{c_i}\Big)$ and sends $\mathsf{pp}, \mathsf{ct}^*, k^*$ to the adversary. The adversary outputs a bit γ'. $\mathcal{B}$ outputs γ' as its guess in n-DBDHI game.

We now analyze the advantage of $\mathcal{B}$ in n-DBDHI game. As β is sampled uniformly, α is also uniformly distributed. As $\beta \neq 0 \mod p$ and ρ is uniformly distributed, $h^{\rho/\beta}$ is also uniformly distributed in $\mathbb{G}_1$. If $\gamma = 0$, then $(\mathsf{pp}, \mathsf{ct}^*, k^*)$ is same as $\Big(\mathsf{pp}, E_1(\mathsf{pp}, (j, 1-x_j); \rho'), E_2(\mathsf{pp}, (f(\mathsf{pp}, x), j, 1-x_j); \rho')\Big)$. If $\gamma = 1$, then k^* is uniformly random. As $\mathcal{A}$ distinguishes these 2 distributions with non-negligible probability, $|\Pr[\gamma' = 1 | \gamma = 0] - \Pr[\gamma' = 1 | \gamma = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks n-DBDHI assumption. ■

Smoothness. We now prove that the above construction satisfies (k, n)-smoothness property for any $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$.

Lemma 5.3.4. *The above construction satisfies (k, n)-smoothness property for any $k \geq m + 2\lambda$ and $n \leq k + m - 2\lambda$ as per Definition 2.1.3.*

Proof. We prove the theorem via a sequence of the following hybrids.

Hybrid H_0: This is the same as the original smoothness security game.

1. The adversary sends two (k, n)-sources S_0 and S_1 to the challenger. The challenger samples a group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_T, e, p) \leftarrow \mathsf{Setup}(1^\lambda)$, a generator $g \leftarrow \mathbb{G}_1$ and exponents $d_0, d_1 \leftarrow \mathbb{Z}_p$.
2. It then samples exponent $\alpha \leftarrow \mathbb{Z}_p^*$ and computes $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.
3. The challenger samples a bit $b \leftarrow \{0, 1\}$, a string $x \leftarrow S_b$ and sends $\mathsf{pp}, y = g^{(d_0 x + d_1) \cdot \prod_{j=1}^{n} (\alpha + 2j + x_j)}$ to the adversary.
4. The adversary outputs a bit b'.

Hybrid H_1: In this hybrid, the challenger samples α in public parameters from $[1, p - 2n - 2]$ instead of $\mathbb{Z}_p^*$

2. It then samples exponent $\alpha \leftarrow [1, p - 2n - 2]$ and computes $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.

Hybrid H_2: In this hybrid, the challenger samples the challenge y uniformly at random.

3. The challenger samples $y \leftarrow \mathbb{G}_1$ and sends pp, y to the adversary.

For any adversary $\mathcal{A}$, let the probability that $b' = b$ in Hybrid H_s be $p_s^{\mathcal{A}}$. We know that, $p_2^{\mathcal{A}} = 1/2$ as y is independent of b. We prove that for every PPT adversary $\mathcal{A}$, $|p_0^{\mathcal{A}} - p_2^{\mathcal{A}}|$ is negligible.

Claim 5.3.1. *For every adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}| \leq negl(\lambda)$.*

Proof. The distribution of challenger's output is same in Hybrids H_0 and H_1, except when $\alpha \in [p-1, p-2n-1]$. This event happens with probability $(2n+1)/p$. Assuming p is super-polynomial in λ, the event $\alpha \in [p-1, p-2n-1]$ happens with negligible probability. ■

Claim 5.3.2. *For every adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}| \leq negl(\lambda)$.*

Proof. As the minimum entropy of the distribution $\{x : b \leftarrow \{0,1\}, x \leftarrow S_b\}$ is $k \geq \log p + 2\lambda$ and as α is sampled from $[1, p-2n-2]$, by Lemma 5.3.1, $(d_0 x + d_1) \cdot \prod_{j=1}^{n}(\alpha + 2j + x_j)$ for $x \leftarrow S_b$ is indistinguishable from uniform distribution on $\mathbb{Z}_p$. ■

By the above claims and triangle inequality, any adversary's advantage in the original smoothness game H_0 is negligible. ■

5.4 Hinting PRG based on Φ-Hiding Assumption

In this section, we provide our (n, ℓ)-hinting PRG (HPRG) construction based on the Φ-hiding assumption. For an RSA modulus N and parameters

λ, ℓ, let $\mathsf{Ext} : \mathbb{Z}_N \times \mathbb{S} \to \{0,1\}^\ell$ be a $(\lambda - 1, \epsilon_{\mathsf{ext}})$ strong extractor, where ϵ_{ext} is negligible in the parameter λ.[3] Below we describe our construction.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, n)$. The setup algorithm takes as input the security parameter λ. It sets RSA modulus bit length $\kappa = 5\lambda$ and number of blocks $n = \kappa + 2\lambda$.

Next, it samples modulus N as $N \leftarrow \mathsf{RSA}(\kappa)$, seed $\mathfrak{s} \leftarrow \mathbb{S}$, generator $g \leftarrow \mathbb{Z}_N^*$, $2n$ (λ-bit) primes $\{e_{i,b}\}_{i,b}$ as $e_{i,b} \leftarrow \mathrm{PRIMES}(\lambda)$ for $(i,b) \in [n] \times \{0,1\}$, and elements $d_0, d_1 \leftarrow \mathbb{Z}_N$. Finally, it outputs the public parameters pp as $\mathsf{pp} = \left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$.

$\mathsf{Eval}(\mathsf{pp}, x, j) \to y$. Let $\mathsf{pp} = \left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$. The evaluation algorithm proceeds as follows. Let $\widetilde{e_1} = \lceil \log_2 N \rceil$, $\widetilde{e_2} = \lceil \log_3 N \rceil$, and $f_1 = 2^{\widetilde{e_1}}$, $f_2 = 3^{\widetilde{e_2}}$.

It computes $h = g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_{i \in [n] \setminus \{j\}} e_{i,x_i}} \pmod N$, and outputs $y = \mathsf{Ext}(h, \mathfrak{s})$.

5.4.1 Optimization by Sharing Computation

A naïve method of computing $\mathsf{Eval}(\mathsf{pp}, x, j)$ for all indices j involves $O(n^2)$ exponentiations. However, a significant amount of these exponentiations are redundant. This is because for any indices $j_1 \neq j_2$, most of the operations involved in computing $\mathsf{Eval}(\mathsf{pp}, x, j_1)$ and $\mathsf{Eval}(\mathsf{pp}, x, j_2)$ are same.

[3]Note that such an extractor exists when $\ell < c\lambda$ for any fixed constant c. The Hinting PRG construction can be extended for $\ell \geq \lambda$ by using standard PRG.

We now describe an efficient procedure that computes $\mathsf{Eval}(\mathsf{pp}, x, j)$ for all indices j using only $O(n \log n)$ exponentiations. We use the technique of dynamic programming to achieve optimization.

Consider the recursive procedure described in Algorithm 1. The procedure takes as input a base generator h, an integer n, a list of exponents exp of length n and RSA modulus N. It outputs a list $[h^{\prod_{i \neq j} \mathsf{exp}_i} \mod N]_{j \in [n]}$. In order to compute $\mathsf{Eval}(\mathsf{pp}, x, j)$ for all j, we invoke the procedure $\mathsf{ComputeHPRG}$ on generator $h = g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)}$ and exponents $\mathsf{exp} = [e_{i,x_i}]_{i \in [n]}$. The procedure $\mathsf{ComputeHPRG}$ divides the list of exponents into groups of 2, multiplies the exponents in each group and obtains the list subexp. It then recurses on the smaller list of exponents subexp, and then uses its output to compute $[h^{\prod_{i \neq j} \mathsf{exp}_i} \mod N]_{j \in [n]}$.

Let us now analyze the total time taken by the algorithm. Initially the $\mathsf{ComputeHPRG}$ procedure is invoked on the list exp consisting of n λ-bit exponents $[e_{i,x_i}]_{i \in [n]}$. The procedure involves n exponentiations with λ bit exponents (Lines 8-10) and calls the $\mathsf{ComputeHPRG}$ procedure recursively on $n/2$ 2λ-bit exponents subexp (Multiplying numbers as in Line 5 is asymptotically faster than exponentiation and therefore we ignore the time required for this operation for the sake of simplicity). Similarly, the i^{th} recursively call to the $\mathsf{ComputeHPRG}$ procedure involves $n/2^i$ exponentiations with $2^i \cdot \lambda$ bit exponents. As the computational effort required for this is the same as the performing n exponentiations with λ bit exponents, and as there can be at most $\log n$ recursive calls, the algorithm totally involves $O(n \log n)$ exponentiations

with λ bit exponents.

Algorithm 1 Recursive procedure for computing HPRG

procedure ComputeHPRG(generator h, int n, list exp, int N)
  if $n = 1$ **then return** $[h]$
  else if $n = 2$ **then return** $[h^{\mathsf{exp}_2} \mod N,\ h^{\mathsf{exp}_1} \mod N]$
  else
    $\mathsf{subexp} \leftarrow [\mathsf{exp}_{2\cdot i-1} \cdot \mathsf{exp}_{2\cdot i}]_{1 \le i \le \lfloor n/2 \rfloor}$
    if $n \mod 2 = 1$ **then** $\mathsf{subexp} \leftarrow \mathsf{subexp} || \mathsf{exp}_n$
    $\mathsf{suboutput} \leftarrow \text{ComputeHPRG}(h, \lceil n/2 \rceil, \mathsf{subexp})$
    for $1 \le i \le 2 \cdot \lfloor n/2 \rfloor$ **do**
      if $i \mod 2 = 0$ **then** $\mathsf{output}_i = \mathsf{suboutput}_{\lceil i/2 \rceil}^{\mathsf{exp}_{i-1}} \mod N$
      else if $i \mod 2 = 1$ **then** $\mathsf{output}_i = \mathsf{suboutput}_{\lceil i/2 \rceil}^{\mathsf{exp}_{i+1}} \mod N$
    if $n \mod 2 = 1$ **then** $\mathsf{output} \leftarrow \mathsf{output} || \mathsf{suboutput}_{\lceil n/2 \rceil}$
    return output

5.4.2 Security

We prove that the construction described above is a secure HPRG. Formally, we prove the following.

Theorem 5.4.1. *If the Φ-hiding assumption (Assumption 1) holds, then the HPRG construction described above is secure as per Definition 2.2.1.*

Proof. First, we introduce some useful notations. For any sequence $\{e_{i,b}\}_{i,b}$ and string $x \in \{0,1\}^n$, we use $\mathbf{e}_x$ as a shorthand for the subset product $\prod_{i=1}^{n} e_{i,x_i}$. Let p_i denote the i^{th} (smallest) prime, i.e. $p_1 = 2, p_2 = 3, \ldots$, and $\widetilde{e_i} = \lceil \log_{p_i} N \rceil$ for all i. And, let f_i denote $p_i^{\widetilde{e_i}}$ for all i. For any constant $\epsilon > 0$, let j_ϵ be the smallest index such that $p_{j_\epsilon} > (2\sqrt{2} \log N/\epsilon)^3$. Now we describe our proof.

The proof of security follows via a sequence of hybrids. Below we first describe the sequence of hybrids and later argue indistinguishability to complete the proof. At a very high level, the proof structure is somewhat similar to that used in [104], where for proving security, one first assumes (for the sake of contradiction) that the adversary wins with some non-negligible probability δ and then, depending upon δ, one could describe a sequence of hybrids such that no PPT adversary can win with probability more than δ. This acts as a contradiction, thereby completing the proof.

Intuitively, the main idea is to first switch the randomly sampled terms $y^0_{j,1-x_j}$ to be instead sampled in a very structured way where $y^0_{j,1-x_j} = \mathsf{Ext}(h_{j,1-x_j}, \mathfrak{s})$ and $h_{j,1-x_j} = g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) e^{-1}_{j,1-x_j} \prod_{i \in [n]} e_{i,x_i}}$. This follows from the Φ-hiding based extractor lemma (Lemma 4.4.1). Next, using our new hashing lemma (Theorem 4.2.1), we now instead of computing $g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_{i \in [n]} e_{i,x_i}}$, sample $z \leftarrow \mathbb{Z}^*_N$ and compute $z^{\prod_{k=1}^{j_\epsilon} f_k}$. In the next hybrid, we replace challenge with random elements in $\mathbb{Z}^*_N$ using the Φ-hiding based extractor lemma (Lemma 4.4.1). Concretely, Hybrid 1 corresponds to HPRG game where the challenger always chooses $\beta = 0$, i.e. samples half of the challenge matrix as appropriate functions of the seed x and remaining randomly. Hybrid 4 corresponds to HPRG game where the challenger always chooses $\beta = 1$, i.e. challenge matrix consists of random entries.

For any PPT adversary $\mathcal{A}$, let the probability that $\mathcal{A}$ outputs 1 in Hybrid t be denoted as $p^{\mathcal{A}}_t$. For the sake of contradiction, we assume that the adversary $\mathcal{A}$ wins with non-negligible probability $\delta(\lambda)$, which suggests that

$p_1^{\mathcal{A}} - p_4^{\mathcal{A}} = \delta(\lambda)$. This means that there exists a polynomial $v(\cdot)$ such that $\delta(\lambda) \geq 1/v(\lambda)$ infinitely often for $\lambda \in \mathbb{N}$. Let $\epsilon(\lambda) = 1/v(\lambda)$. We will drop the dependence on λ whenever clear from the context.[4]

Hybrid 1. This is same as the original HPRG game, where the challenger always chooses $\beta = 0$.

1. The challenger sets $\kappa = 5\lambda$, $n = \kappa + 2\lambda$. It samples the public parameters $\mathsf{pp} = \left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$ as:

$$N \leftarrow \mathsf{RSA}(\kappa), \quad d_0, d_1 \in \mathbb{Z}_N, \quad g \in \mathbb{Z}_N^*, \quad \mathfrak{s} \leftarrow \mathbb{S},$$
$$e_{i,b} \leftarrow \text{PRIMES}(\lambda) \ (\forall \ (i,b) \in [n] \times \{0,1\}).$$

2. Next, it samples a random HPRG seed $x \leftarrow \{0,1\}^n$, computes the HPRG challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$y_0 = \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x}, \mathfrak{s}),$$
$$\forall \ i \in [n], \quad y_{i,x_i} = \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x e_{i,x_i}^{-1}}, \mathfrak{s}),$$
$$\forall \ i \in [n], \quad y_{i,1-x_i} \leftarrow \{0,1\}^\ell.$$

3. The challenger sends public parameters pp, n and the challenge $(y_0, \{y_{i,b}\}_{i,b})$ to the adversary. Finally, the adversary outputs a bit β'.

[4]Note that, throughout the analysis, we provide a non-uniform reduction where the description of hybrids and the reduction algorithm depends upon ϵ. Note that one could instead avoid such a non-uniform choice by first running the adversary sufficiently many times to estimate the advantage ϵ as $\widetilde{\epsilon}$, and later on, use the estimated advantage $\widetilde{\epsilon}$ instead. Therefore, for ease of exposition, we give a non-uniform reduction.

Hybrid $1.i^*$ ($i^* \in [n]$). This hybrid is same as hybrid 1, except that the challenger chooses $y_{i,1-x_i}$ in a structured way for all $i \leq i^*$.

2. Next, it samples a random HPRG seed $x \leftarrow \{0,1\}^n$, computes the HPRG challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$\begin{aligned} y_0 &= \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x}, \mathfrak{s}), \\ \forall\ i \in [n], \quad y_{i,x_i} &= \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x e_{i,x_i}^{-1}}, \mathfrak{s}), \\ \forall\ i \in [i^*], \quad y_{i,1-x_i} &= \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x e_{i,1-x_i}^{-1}}, \mathfrak{s}), \\ \forall\ i \in [n] \setminus [i^*], \quad y_{i,1-x_i} &\leftarrow \{0,1\}^\ell. \end{aligned}$$

Note that the challenger knows $\phi(N)$, thus it can compute all the necessary inverses.

Hybrid 2. This hybrid is similar to Hybrid $1.n$, except none of the challenge terms have any dependence on the HPRG seed x.

1. The challenger sets $\kappa = 5\lambda$, $n = \kappa + 2\lambda$. It samples the public parameters $\mathsf{pp} = \left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$ as:

$$N \leftarrow \mathsf{RSA}(\kappa), \quad d_0, d_1 \in \mathbb{Z}_N, \quad \mathfrak{s} \leftarrow \mathbb{S}, \quad h \in \mathbb{Z}_N^*, \quad g = h^{\prod_{k=3}^{j_\epsilon} f_k}.$$

$$e_{i,b} \leftarrow \mathrm{PRIMES}(\lambda)\ (\forall\ (i,b) \in [n] \times \{0,1\})$$

2. Next, it samples a random HPRG seed $x \leftarrow \{0,1\}^n$, computes the HPRG

challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$\begin{aligned} & z \leftarrow \mathbb{Z}_N^*, \\ & y_0 = \mathsf{Ext}(z^{\prod_{k=1}^{j_\epsilon} f_k}, \mathfrak{s}), \\ \forall\ i \in [n], b \in \{0,1\}, \quad & y_{i,b} = \mathsf{Ext}(z^{e_{i,b}^{-1} \prod_{k=1}^{j_\epsilon} f_k}, \mathfrak{s}). \end{aligned}$$

Hybrid $3.i^*.b^*$ $(i^* \in [n], b^* \in \{0,1\})$. This hybrid is similar to Hybrid 2, except that the challenger chooses $y_{i,b}$ randomly for all $(i,b) \preceq (i^*, b^*)$.[5]

2. Next, it samples a random HPRG seed $x \leftarrow \{0,1\}^n$, computes the HPRG challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$\begin{aligned} & z \leftarrow \mathbb{Z}_N^*, \\ & y_0 = \mathsf{Ext}(z^{\prod_{k=1}^{j_\epsilon} f_k}, \mathfrak{s}), \\ \forall\ (i,b) \preceq (i^*, b^*), \quad & y_{i,b} \leftarrow \{0,1\}^\ell, \\ \forall\ (i,b) \succ (i^*, b^*), \quad & y_{i,b} = \mathsf{Ext}(z^{e_{i,b}^{-1} \prod_{k=1}^{j_\epsilon} f_k}, \mathfrak{s}). \end{aligned}$$

Hybrid 4. This hybrid is similar to Hybrid $3.n.1$ except that y_0 is also generated randomly.

2. Next, it samples a random HPRG seed $x \leftarrow \{0,1\}^n$, computes the HPRG

[5] Here and throughout this section, $\preceq$ is a shorthand for the following relation: $(i,b) \preceq (i^*, b^*) \equiv i < i^* \vee (i = i^* \wedge b \leq b^*)$. And, $\succ$ is analogously defined, but as the converse of $\preceq$.

challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$y_0 \leftarrow \{0,1\}^\ell,$$
$$\forall\ i \in [n], b \in \{0,1\}, \quad y_{i,b} \leftarrow \{0,1\}^\ell.$$

Now, we argue indistinguishability between the hybrids described above. Below we use Hybrid 1.0 to correspond to Hybrid 1, and Hybrid 3.0.1 to correspond to Hybrid 2.

Lemma 5.4.1. *Assuming the Φ-hiding assumption holds, then for every PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $i^* \in [n]$, $p^{\mathcal{A}}_{1.(i^*-1)} - p^{\mathcal{A}}_{1.i^*} \leq negl(\lambda)$.*

Proof. The proof of this lemma follows directly from our Φ-hiding-based extractor lemma (Lemma 4.4.1). Suppose that $\mathcal{A}$ distinguishes between Hybrids $1.(i^*-1)$ and $1.i^*$ with non-negligible probability γ, for some $i^* \in [n]$. We use $\mathcal{A}$ to build a reduction algorithm $\mathcal{B}$ that violates our Φ-hiding-based extractor lemma, thereby leading us to a contradiction. Below we provide more details.

The reduction algorithm $\mathcal{B}$ first receives the parameters $(N, \mathfrak{s}, e, \widetilde{g})$ where e is a λ-bit prime and $\widetilde{g}$ is a random element in $\mathbb{Z}_N^*$. $\mathcal{B}$ then samples parameters d_0, d_1 and HPRG seed x randomly as in Hybrid 1. It also samples primes $e_{i,b}$ for all $(i,b) \neq (i^*, 1-x_{i^*})$ as in Hybrid 1. It sets $e_{i^*,1-x_{i^*}}, g$ as $e_{i^*,1-x_{i^*}} = e$ and $g = \widetilde{g}^{\prod_{i<i^*} e_{i,1-x_i}}$. Now it sets the public parameters pp as $\mathsf{pp} = \left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$. Next, $\mathcal{B}$ sets exponent F as $F = f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x$. If $e \mid F$, then $\mathcal{B}$ aborts and guesses randomly, otherwise it sends F to the

challenger and continues. The challenger responds with a challenge bit string y, and then $\mathcal{B}$ computes the HPRG challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$\begin{aligned}
& y_0 = \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x}, \mathfrak{s}), \\
\forall\ i \in [n], \quad & y_{i,x_i} = \mathsf{Ext}(g^{f_1 \cdot f_2 \cdot (d_0 x + d_1) \prod_{j \neq i} e_{j,x_j}}, \mathfrak{s}), \\
\forall\ i \in [i^* - 1], \quad & y_{i,1-x_i} = \mathsf{Ext}(\widetilde{g}^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x \prod_{j < i^* \wedge j \neq i} e_{j,1-x_j}}, \mathfrak{s}), \\
& y_{i^*,1-x_{i^*}} = y, \\
\forall\ i \in [n] \setminus [i^*], \quad & y_{i,1-x_i} \leftarrow \{0,1\}^{\ell}.
\end{aligned}$$

Finally, $\mathcal{B}$ sends pp, n and $(y_0, \{y_{i,b}\}_{i,b})$ to $\mathcal{A}$, and $\mathcal{A}$ outputs its guess β'. If $\beta' = 1$, then $\mathcal{B}$ outputs 0 (i.e., y is computed honestly) as its guess, otherwise it outputs 1 (i.e., y is random bit string) as its guess.

Let us now analyze the advantage of the reduction algorithm $\mathcal{B}$. First, note that $\mathcal{B}$ samples d_0, d_1, x and $e_{i,b}$ for $(i,b) \neq (i^*, 1 - x_{i^*})$ randomly (from appropriate distributions). Therefore, we have that the event $e \mid F$ occurs with only negligible probability since e is a random λ-bit prime, thus $\mathcal{B}$ aborts with atmost negligible probability. Second, since $e_{i,1-x_i}$ are randomly drawn λ-bit primes for $i \neq i^*$, thus sampling g as $g = \widetilde{g}^{\prod_{i<i^*} e_{i,1-x_i}}, \widetilde{g} \leftarrow \mathbb{Z}_N^*$ instead of $g \leftarrow \mathbb{Z}_N^*$ is statistically indistinguishable. Therefore, $\mathcal{B}$ simulates one of hybrids $1.(i^* - 1)$ and $1.i^*$ (in a statistically indistinguishable way) depending upon whether the challenge y is computed as $y = \mathsf{Ext}(\widetilde{g}^{F/e}, \mathfrak{s})$ or $y \leftarrow \{0,1\}^{\ell}$. Hence, if $\mathcal{A}$ distinguishes with non-negligible probability γ, then $\mathcal{B}$'s advantage is also negligibly close to γ. Thus, the lemma follows. ∎

Lemma 5.4.2. *Assuming the Φ-hiding assumption holds, then for every PPT adversary $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that for all $\lambda \in \mathbb{N}$, satisfying $\delta(\lambda) \geq \epsilon(\lambda) = 1/v(\lambda)$, $p^{\mathcal{A}}_{1.n} - p^{\mathcal{A}}_{2} \leq \epsilon(\lambda)/2 + negl(\lambda)$.*

Proof. This lemma's proof follows directly from our new hashing lemma (Theorem 4.2.1). Suppose that $\mathcal{A}$ distinguishes between Hybrids $1.n$ and 2 with probability $\epsilon/2 + \gamma$, where γ is non-negligible. We use $\mathcal{A}$ to build a reduction algorithm $\mathcal{B}$ that violates the security requirement of our new hash lemma, thereby leading us to a contradiction. Below we provide more details.

The reduction algorithm $\mathcal{B}$ first receives the parameters $(N, \widetilde{g}, K, y)$, where $K = (d_0, d_1, \{e_{i,b}\}_{i,b})$. $\mathcal{B}$ then samples an extractor seed $\mathfrak{s}$ randomly. It computes exponent $E = \prod_{i,b} e_{i,b}$ and generator $g = \widetilde{g}^E$, and it sets the public parameters pp as $\mathsf{pp} = \left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$. Next, $\mathcal{B}$ computes the HPRG challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$y_0 = \mathsf{Ext}(y^E, \mathfrak{s}),$$
$$\forall\ i \in [n], b \in \{0,1\}, \quad y_{i,b} = \mathsf{Ext}(y^{\prod_{(j,c)\neq(i,b)} e_{j,c}}, \mathfrak{s}).$$

Finally, $\mathcal{B}$ sends pp, n and $(y_0, \{y_{i,b}\}_{i,b})$ to $\mathcal{A}$, and $\mathcal{A}$ outputs its guess β'. If $\beta' = 1$, then $\mathcal{B}$ outputs 0 (i.e., y is computed honestly) as its guess, otherwise it outputs 1 (i.e., y is random bit string) as its guess.

Let us now analyze the advantage of the reduction algorithm $\mathcal{B}$. First, note that the challenger samples all the primes $e_{i,b}$'s randomly, thus we have that with all-but-negligible probability, all $e_{i,b}$'s are co-prime w.r.t. $\phi(N)$.

Therefore, sampling g as $g = \widetilde{g}^E, \widetilde{g} \leftarrow \mathbb{Z}_N^*$ instead of $g \leftarrow \mathbb{Z}_N^*$ is statistically indistinguishable. Similarly, sampling g as $g = \widetilde{g}^E, \widetilde{g} = h^{\prod_{k=3}^{j_\epsilon} f_k}, h \leftarrow \mathbb{Z}_N^*$ instead of $g = \widetilde{g}^{\prod_{k=3}^{j_\epsilon} f_k}, \widetilde{g} \leftarrow \mathbb{Z}_N^*$ is statistically indistinguishable. Therefore, $\mathcal{B}$ simulates one of hybrids $1.n$ and 2 (in a statistically indistinguishable way) depending upon whether the challenge y is computed as $y = \widetilde{g}^{f_1 \cdot f_2 \cdot (d_0 x + d_1)\mathbf{e}_x}$ for a random x, or $y = z^{\prod_{k=1}^{j_\epsilon} f_k}$ for a random $z \in \mathbb{Z}_N^*$. Hence, if $\mathcal{A}$ distinguishes with non-negligible probability $\epsilon/2 + \gamma$, then $\mathcal{B}$'s advantage is also negligibly close to $\epsilon/2 + \gamma$. Thus, the lemma follows. ■

Lemma 5.4.3. *Assuming the Φ-hiding assumption holds, then for every PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $i^* \in [n]$, $p^{\mathcal{A}}_{3.i^*.0} - p^{\mathcal{A}}_{3.i^*.1} \leq negl(\lambda)$.*

Proof. The proof of this lemma follows directly from our Φ-hiding-based extractor lemma (Lemma 4.4.1). It is quite similar to the proof of Lemma 5.4.1. Suppose that $\mathcal{A}$ distinguishes between Hybrids $3.i^*.0$ and $3.i^*.1$ with non-negligible probability γ, for some $i^* \in [n]$. We use $\mathcal{A}$ to build a reduction algorithm $\mathcal{B}$ that violates our Φ-hiding-based extractor lemma, thereby leading us to a contradiction. Below we provide more details.

The reduction algorithm $\mathcal{B}$ first receives the parameters $(N, \mathfrak{s}, e, \widetilde{g})$ where e is a λ-bit prime and $\widetilde{g}$ is a random element in $\mathbb{Z}_N^*$. $\mathcal{B}$ then samples parameters d_0, d_1 and HPRG seed x randomly as in Hybrid 2. It also samples primes $e_{i,b}$ for all $(i, b) \neq (i^*, 1)$ as in Hybrid 2. It sets $e_{i^*,1}$ as $e_{i^*,1} = e$ and samples $g \leftarrow \mathbb{Z}_N^*$. Now it sets the public parameters pp as $\mathsf{pp} =$

$\left(N, \mathfrak{s}, g, \{e_{i,b}\}_{i,b}, d_0, d_1\right)$. Next, $\mathcal{B}$ sets exponent F as $F = \prod_{k=1}^{j_\epsilon} f_k \prod_{i>i^*, b\in\{0,1\}} e_{i,b}$. If $e \mid F$, then $\mathcal{B}$ aborts and guesses randomly, otherwise it sends F to the challenger and continues. The challenger responds with a challenge bit string y, and then $\mathcal{B}$ computes the HPRG challenge $(y_0, \{y_{i,b}\}_{i,b})$ as:

$$
\begin{aligned}
& & z &= \widetilde{g}^{\prod_{i>i^*, b\in\{0,1\}} e_{i,b}}, \\
& & y_0 &= \mathsf{Ext}(z^{\prod_{k=1}^{j_\epsilon} f_k}, \mathfrak{s}), \\
\forall\ (i,b) \preceq (i^*, 0), & & y_{i,b} &\leftarrow \{0,1\}^\ell, \\
& & y_{i^*,1} &= y, \\
\forall\ (i,b) \succ (i^*, 1), & & y_{i,b} &= \mathsf{Ext}(\widetilde{g}^{\prod_{k=1}^{j_\epsilon} f_k \prod_{(j,c)\neq(i,b)\wedge j>i^*, c\in\{0,1\}} e_{j,c}}, \mathfrak{s}).
\end{aligned}
$$

Finally, $\mathcal{B}$ sends pp, n and $(y_0, \{y_{i,b}\}_{i,b})$ to $\mathcal{A}$, and $\mathcal{A}$ outputs its guess β'. If $\beta' = 1$, then $\mathcal{B}$ outputs 0 (i.e., y is computed honestly) as its guess, otherwise it outputs 1 (i.e., y is random bit string) as its guess.

Let us now analyze the advantage of the reduction algorithm $\mathcal{B}$. First, note that $\mathcal{B}$ samples $e_{i,b}$ for $(i,b) \neq (i^*, 1)$ randomly, thus we have that the event $e \mid F$ occurs with only negligible probability since e is a random λ-bit prime. So, $\mathcal{B}$ aborts with atmost negligible probability. Second, since $e_{i,b}$ are randomly drawn λ-bit primes for $i > i^*$, thus sampling z as $z = \widetilde{g}^{\prod_{i>i^*, b} e_{i,b}}, \widetilde{g} \leftarrow \mathbb{Z}_N^*$ instead of $z \leftarrow \mathbb{Z}_N^*$ is statistically indistinguishable. Therefore, $\mathcal{B}$ simulates one of hybrids $3.i^*.0$ and $3.i^*.1$ (in a statistically indistinguishable way) depending upon whether the challenge y is computed as $y = \mathsf{Ext}(\widetilde{g}^{F/e}, \mathfrak{s})$ or $y \leftarrow \{0,1\}^\ell$. Hence, if $\mathcal{A}$ distinguishes with non-negligible

probability γ, then $\mathcal{B}$'s advantage is also negligibly close to γ. Thus, the lemma follows. ■

Lemma 5.4.4. *Assuming the Φ-hiding assumption holds, then for every PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $i^* \in [n]$, $p^{\mathcal{A}}_{3.(i^*-1).1} - p^{\mathcal{A}}_{3.i^*.0} \leq negl(\lambda)$.*

Proof. The proof of this lemma is identical to the proof of Lemma 5.4.3. ■

Lemma 5.4.5. *For every adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p^{\mathcal{A}}_{3.n.1} - p^{\mathcal{A}}_4 \leq negl(\lambda)$.*

Proof. This follows directly from the strong extractor guarantee of Ext. Let $S \subset \mathbb{Z}^*_N$ denote the set $\left\{ z \in \mathbb{Z}^*_N : z = \widetilde{z}^{\prod_{k=1}^{j_\epsilon} f_k} \wedge \widetilde{z} \in \mathbb{Z}^*_N \right\}$. Now since N is an RSA modulus of bit length κ, we have that $\log_2 |S| \geq \lambda$ with all-but-negligible probability. Note that y_0 is sampled as $y_0 = \mathsf{Ext}(z, \mathfrak{s})$ where $z \leftarrow S$. Thus, by extractor security, given $\log_2 |S| \geq \lambda$, the claim follows since ϵ_{ext} is negligible. ■

Combining Lemmas 5.4.1 to 5.4.5, we get that $p^{\mathcal{A}}_1 - p^{\mathcal{A}}_4 \leq \epsilon(\lambda)/2 + \widetilde{\mathrm{negl}}(\lambda)$ for some negligible function $\widetilde{\mathrm{negl}}(\cdot)$ (whenever $\delta(\lambda) \geq \epsilon(\lambda)$). Thus, we can conclude that $p^{\mathcal{A}}_1 - p^{\mathcal{A}}_4 \leq 2\epsilon(\lambda)/3$ infinitely often. This contradicts our assumption that $p^{\mathcal{A}}_1 - p^{\mathcal{A}}_4 \geq \delta(\lambda) \geq \epsilon(\lambda)$. Thus, this completes the proof.

■

5.5 Hinting PRG from q-DDHI Assumption

In this section, we construct (n, ℓ)-hinting PRG from any $2n$-DDHI hard group generator GGen. Suppose $\mathsf{GGen}(1^\lambda)$ generates a group of order at most 2^κ. The below construction requires $n \geq k + 2\lambda$ and $k = \omega(\log \lambda)$. We construct a hinting PRG that outputs elements in a group for simplicity. The construction can be extended to output ℓ-length bit strings for any polynomial ℓ by using standard PRGs and randomness extractors.

$\mathsf{Setup}(1^\lambda)$: Sample a group $\mathcal{G} = (\mathbb{G}, p) \leftarrow \mathsf{GGen}(1^\lambda)$. Sample a generator $g \leftarrow \mathbb{G}$ and random exponents $\alpha \leftarrow \mathbb{Z}_p^*$ and $d_0, d_1 \leftarrow \mathbb{Z}_p$. Output the public parameters $(\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.

$\mathsf{Eval}(\mathsf{pp}, x, i)$: Parse public parameters pp as $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$. Set $f = g^{(d_0 x + d_1) \cdot \prod_{j \in [1,n] \setminus \{i\}} (\alpha + 2j + x_j)}$. Expand the polynomial f as $\prod_{j=0}^{n} c_j \alpha^j$. Compute and output $\prod_{j=0}^{n} \left(g^{\alpha^j}\right)^{c_j}$.

5.5.1 Security

We now prove that the above construction is a secure Hinting PRG. Formally, we prove

Theorem 5.5.1. *If the $2n$-DDHI assumption (Assumption 3) holds on group generator GGen, then the hinting PRG construction described above is secure as per Definition 2.2.1.*

Proof. We now prove that the above construction is a secure hinting PRG via a sequence of hybrids. In the proof, we use F_x as a shorthand for $\prod_{j=1}^{n} (\alpha + 2j +$

x_j). In the first hybrid, y_{i,x_i} are structured ($y_{i,x_i} = g^{(d_0x+d_1)\cdot(\alpha+2i+x_i)^{-1}\prod_{j=1}^{n}(\alpha+2j+x_j)}$) and $y_{i,1-x_i}$ are sampled randomly in the HPRG challenge. In the next hybrid, we switch $y_{i,1-x_i}$ to structured values, i.e., $y_{i,b} = g^{(d_0x+d_1)\cdot(\alpha+2i+b)^{-1}\prod_{j=1}^{n}(\alpha+2j+x_j)}$ using DDHI assumption. In the next hybrid, we erase the information about x in the challenge. Concretely, we show that $(d_0x + d_1) \cdot \prod_{j=1}^{n}(\alpha + 2j + x_j)$ mod p is statistically indstinguishable from random value in $\mathbb{Z}_p$ and consequently switch $y_{i,b}$ to $g^{r\cdot(\alpha+2i+b)^{-1}}$ for a randomly sampled r. We then use DDHI assumption to switch the challenge to random group elements.

Hybrid H_0: This is same as the original hinting PRG game when the challenger always chooses $\beta = 0$.

1. The challenger first samples a group $\mathcal{G} = (\mathbb{G}, p) \leftarrow \mathsf{GGen}(1^\lambda)$ and generator $g \leftarrow \mathbb{G}$. It then samples random values $\alpha, d_0, d_1 \leftarrow \mathbb{Z}_p$. It then computes $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.
2. It then samples a bit string $x \leftarrow \{0,1\}^n$, random values $r_i \leftarrow \mathbb{Z}_p$ for $i \in [n]$. It then computes the challenge $y_0 = g^{F_x}, y_{i,x_i} = g^{F_x/(\alpha+2i+x_i)}, y_{i,1-x_i} = g^{r_i}$ for $i \in [n]$.
3. The challenger sends public parameters pp and the challenge $\{y_0, \{y_{i,b}\}_{i,b}\}$ to the adversary.
4. The adversary outputs a bit β'.

Hybrid H_1: This is same as previous game, except that the challenger aborts if there exists an $i \in [2n+1]$ s.t. $\alpha + i = 0 \mod p$.

1. The challenger first samples a group $\mathcal{G} = (\mathbb{G}, p) \leftarrow \mathsf{GGen}(1^\lambda)$ and generator $g \leftarrow \mathbb{G}$. It then samples random values $\alpha, d_0, d_1 \leftarrow \mathbb{Z}_p$. It aborts if $\alpha + i = 0 \mod p$ for any $i \in [2n+1]$. It then computes $\mathsf{pp} = (p, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.

We now define a sequence of $n+1$ hybrids. For the sake of simplicity, let Hybrid $H_{2.0}$ be same as Hybrid H_1.

Hybrid $H_{2.j}$ $(j \in [n])$: This is same as previous game, except that the challenger computes $y_{i,1-x_i}$ differently.

2. It then samples a bit string $x \leftarrow \{0,1\}^n$, random values $r_i \leftarrow \mathbb{Z}_p$ for $i \in [n]$. It then computes the challenge $y_0 = g^{F_x}$, $y_{i,b} = g^{F_x/(\alpha+2i+b)}$ for all (i,b) s.t. $i \leq j$ or $b = x_i$, $y_{i,1-x_i} = g^{r_i}$ for $i > j$.

Hybrid H_3: This is same as Hybrid $H_{2.n}$ except that the challenger uses a random value r instead of F_x.

2. The challenger samples a random value $r \leftarrow \mathbb{Z}_p$. It then computes the challenge $y_0 = g^r$, $y_{i,b} = g^{r/(\alpha+2i+b)}$ for all $(i,b) \in [n] \times \{0,1\}$.

We next define a sequence of $2n+1$ hybrids. Let us define Hybrid $H_{4.0.1}$ be same as Hybrid H_3.

Hybrid $H_{4.j.b'}$ $(j \in [n], b' \in \{0,1\})$: This is same as Hybrid H_3 except that for $(i,b) \preceq (j,b')$, the challenger samples $y_{i,b}$ uniformly at random.

2. The challenger samples a random value $r \leftarrow \mathbb{Z}_p$ and $r_{i,b} \leftarrow \mathbb{Z}_p$ for $(i,b) \preceq (j,b')$. It then computes the challenge $y_0 = g^r$, $y_{i,b} = g^{r_{i,b}}$ for $(i,b) \preceq (j,b')$, and $y_{i,b} = g^{r/(\alpha+2i+b)}$ for $(i,b) \succ (j,b')$.

Hybrid H_5. This is same as Hybrid $H_{4.n.1}$ except that the challenger does not abort if there exists $i \in [2n+1]$ such that $\alpha + i = 0 \mod p$.

1. The challenger first samples a group $\mathcal{G} = (\mathbb{G}, p) \leftarrow \mathsf{GGen}(1^\lambda)$ and generator $g \leftarrow \mathbb{G}$. It then samples random values $\alpha, d_0, d_1 \leftarrow \mathbb{Z}_p$. It then computes $\mathsf{pp} = (\mathcal{G}, g, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^n}, d_0, d_1)$.

Note that hybrid H_0 is the original hinting PRG game when challenger always chooses $\beta = 0$ and hybrid H_5 is the original hinting PRG game when challenger always chooses $\beta = 1$. We prove that these 2 hybrids are computationally indistinguishable using the following lemmas. For any PPT adversary $\mathcal{A}$, let $p_s^{\mathcal{A}}$ be the probability that $\mathcal{A}$ outputs 1 in Hybrid H_s.

Lemma 5.5.1. *There exists a negligible function $negl(\cdot)$ such that for every adversary $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, we have $|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}| \leq negl(\lambda)$.*

Proof. The distribution of challenger's output is same in Hybrids H_0 and H_1, except when $\alpha \in [p-1, p-2n-1]$. This event happens with probability $(2n+1)/p$. Assuming p is super-polynomial in λ, the event $\alpha \in [p-1, p-2n-1]$ happens with negligible probability. ∎

Lemma 5.5.2. *Assuming $2n$-DDHI assumption holds on group generator* GGen*, for every PPT adversary $\mathcal{A}$ and every index $j \in [n]$, there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{2.j} - p^{\mathcal{A}}_{2.j-1}| \leq negl(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ and an index j s.t. $|p^{\mathcal{A}}_{2.j} - p^{\mathcal{A}}_{2.j-1}|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that breaks $2n$-DDHI assumption on group generator GGen.

The $2n$-DDHI game challenger $\mathcal{C}$ first sends challenge $(\mathcal{G}, h,\ h^{\gamma}, h^{\gamma^2}, \cdots, h^{\gamma^{2n}}, T)$ to the reduction algorithm $\mathcal{B}$. The reduction algorithm samples $x \leftarrow \{0,1\}^n$, implicitly sets $\alpha = \gamma - 2j + x_j - 1$ and aborts if there exists $i \in [2n+1]$ such that $h^{\alpha+i} = \mathbf{1}_{\mathbb{G}}$ ($\mathbf{1}_{\mathbb{G}}$ is identity element of the group $\mathbb{G}$). $\mathcal{B}$ then samples random elements $d_0, d_1 \leftarrow \mathbb{Z}_p$, computes a generator $g = h^{\prod_{k=1}^{j-1}(\alpha+2k+1-x_k)}$, and the public parameters $\mathsf{pp} = (\mathcal{G}, g, g^{\alpha}, g^{\alpha^2}, \cdots g^{\alpha^n}, d_0, d_1)$. $\mathcal{B}$ then computes Hinting PRG challenge as follows. It first computes $y_0 = g^{(d_0 x + d_1)\cdot\prod_{k=1}^{n}(\alpha+2k+x_k)}$, $y_{i,x_i} = g^{(d_0 x + d_1)\cdot\prod_{k\in[1,n]\setminus\{i\}}(\alpha+2k+x_k)}$ for $i \in [n]$. It then computes $y_{i,1-x_i} = h^{(d_0 x + d_1)\cdot\prod_{k\in[j-1]\setminus\{i\}}(\alpha+2k+1-x_k)\cdot\prod_{k=1}^{n}(\alpha+2k+x_k)}$ for $i < j$. It then samples $r_i \leftarrow \mathbb{Z}_p$ and sets $y_{i,1-x_i} = g^{r_i}$ for $i > j$. In order to compute $y_{j,1-x_j}$, $\mathcal{B}$ expands the polynomial

$$f(\alpha) = \frac{(d_0 x + d_1)\cdot\prod_{k=1}^{j-1}(\alpha+2k+1-x_k)\cdot\prod_{k=1}^{n}(\alpha+2k+x_k)}{(\alpha+2j+1-x_j)} = \frac{c}{\gamma} + \sum_{k=0}^{n+j-2} c_k\gamma^k,$$

where $\gamma = (\alpha+2j+1-x_j)$ and $c, \{c_k\}$ are some functions of x, d_0, d_1. $\mathcal{B}$ sets $y_{j,1-x_j} = T^c \cdot \prod_{k=0}^{n+j-2} h^{c_k\gamma^k}$. Note that pp and $(y_0, \{y_{i,b}\}_{i,b})$ can be computed given the challenge sent by $\mathcal{C}$. Finally, $\mathcal{B}$ sends public parameters pp and

challenge $(y_0, \{y_{i,b}\}_{i,b})$ to the adversary $\mathcal{A}$. The adversary outputs a bit β', which $\mathcal{B}$ outputs as its guess in the $2n$-DDHI game.

Note that $\alpha + 2k + 1 - x_k \neq 0 \mod p$ for all $k < j$. Therefore g is a generator of the group $\mathbb{G}$. Moreover, α is uniformly distributed over $\mathbb{Z}_p$ since γ is uniformly sampled from $\mathbb{Z}_p$. Therefore, the distribution of pp sent by $\mathcal{B}$ matches the distribution of pp sent by Hybrid $H_{2.j}$ and $H_{2.j-1}$ challengers. Moreover if T is a random group element, then $\mathcal{B}$ emulates Hybrid $H_{2.j-1}$ challenger to $\mathcal{A}$. If $T = h^{1/\gamma}$, then $\mathcal{B}$ emulates Hybrid $H_{2.j}$ challenger to $\mathcal{A}$. By our assumption, $|p^{\mathcal{A}}_{2.j} - p^{\mathcal{A}}_{2.j-1}| = |\Pr[\beta' = 1|\delta = 0] - \Pr[\beta' = 1|\delta = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks $2n$-DDHI assumption of GGen. ■

Lemma 5.5.3. *For every adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{2.n} - p^{\mathcal{A}}_{3}| \leq negl(\lambda)$.*

Proof. By applying Lemma 5.3.1 with a uniform source S on $\{0,1\}^n$, the statistical difference between Hybrids $H_{2.n}$ and H_3 is negligible in λ. ■

Lemma 5.5.4. *Assuming $2n$-DDHI assumption holds on group generator GGen, for every PPT adversary $\mathcal{A}$, every index $j \in [n]$ and bit b', there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{4.j.b'} - p^{\mathcal{A}}_{4.j-1+b'.1-b'}| \leq negl(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$, an index j and bit b' s.t. $|p^{\mathcal{A}}_{4.j.b'} - p^{\mathcal{A}}_{4.j-1+b'.1-b'}|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that breaks $2n$-DDHI assumption on group generator GGen.

The $2n$-DDHI game challenger $\mathcal{C}$ first sends the challenge $(\mathcal{G}, h, h^{\gamma}, h^{\gamma^2}, \cdots, h^{\gamma^{2n}}, T)$ to the reduction algorithm $\mathcal{B}$. The reduction algorithm samples $x \leftarrow \{0,1\}^n$, implicitly sets $\alpha = \gamma - 2j - b'$ and aborts if there exists $i \in [2n+1]$ such that $h^{\alpha+i} = \mathbf{1}_{\mathbb{G}}$ ($\mathbf{1}_{\mathbb{G}}$ is identity element of the group $\mathbb{G}$). $\mathcal{B}$ then samples random elements $d_0, d_1 \leftarrow \mathbb{Z}_p$, sets the public parameters $\mathsf{pp} = (\mathcal{G}, h, h^{\alpha}, h^{\alpha^2}, \cdots h^{\alpha^n}, d_0, d_1)$. $\mathcal{B}$ then samples $r' \leftarrow \mathbb{Z}_p$, $r_{i,b} \leftarrow \mathbb{Z}_p$ for $(i,b) \prec (j,b')$, and implicitly sets $r = r' \cdot \prod_{\{(k,\beta) \succ (j,b')\}} (\alpha + 2k + \beta)$. $\mathcal{B}$ then computes Hinting PRG challenge as follows. It first computes $y_0 = h^r = h^{r' \cdot \prod_{\{(k,\beta) \succ (j,b')\}} (\alpha+2k+\beta)}$, $y_{i,b} = g^{r_{i,b}}$ for $(i,b) \prec (j,b')$. It then computes $y_{i,b} = h^{r \cdot e_{i,b}^{-1}} = h^{r' \cdot \prod_{(k,\beta) \succ (j,b') \wedge (k,\beta) \neq (i,b)} (\alpha+2k+\beta)}$ for $(i,b) \succ (j,b')$. In order to compute $y_{j,b'}$, $\mathcal{B}$ expands the polynomial

$$f(\alpha) = \frac{r}{(\alpha + 2j + b')} = \frac{r' \cdot \prod_{(k,b) \succ (j,b')} (\alpha + 2k + b)}{(\alpha + 2j + b')} = \frac{c}{\gamma} + \sum_{k=0}^{2n} c_k \gamma^k,$$

where $\gamma = (\alpha + 2j + b')$ and $c, \{c_k\}$ are some functions of r'. $\mathcal{B}$ sets $y_{j,b'} = T^c \cdot \prod_{k=0}^{2n} h^{c_k \gamma^k}$. Note that pp and $(y_0, \{y_{i,b}\}_{i,b})$ can be computed given the challenge sent by $\mathcal{C}$. Finally, $\mathcal{B}$ sends public parameters pp and challenge $(y_0, \{y_{i,b}\}_{i,b})$ to the adversary $\mathcal{A}$. The adversary outputs a bit β', which $\mathcal{B}$ outputs as its guess in $2n$-DDHI game.

Note that the distribution of pp sent by $\mathcal{B}$ matches the distribution of pp sent by Hybrid $H_{4.j.b'}$ and $H_{4.j-1+b'.1-b'}$ challengers. As ρ' is uniformly sampled in $\mathbb{Z}_p$ and $(\alpha+2j+b') \neq 0 \mod p$, ρ is uniformly distributed in $\mathbb{Z}_p$. Moreover, if $T = h^{1/\gamma}$ then $\mathcal{B}$ emulates Hybrid $H_{4.j-1+b'.1.b'}$ challenger to $\mathcal{A}$. If T is a random group element, then $\mathcal{B}$ emulates Hybrid $H_{4.j.b'}$ challenger to $\mathcal{A}$. By

our assumption, $|p^{\mathcal{A}}_{4.j.b'} - p^{\mathcal{A}}_{4.j-1+b'.1-b'}| = |\Pr[\beta' = 1|\delta = 0] - \Pr[\beta' = 1|\delta = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks $2n$-DDHI assumption of $\mathbb{G}$. ■

Lemma 5.5.5. *There exists a negligible function negl$(\cdot)$ such that for every adversary $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{4.n.1} - p^{\mathcal{A}}_{5}| \leq negl(\lambda)$.*

Proof. This proof is the same as proof of Lemma 5.5.1. ■

By the above sequence of lemmas and triangle inequality, Hybrids H_0 and H_5 are computationally indistinguishable. Therefore, the above construction is a secure Hinting PRG. ■

Chapter 6

Performance Evaluation

In this chapter, we discuss how our HPRG and OWFE constructions based on Φ-Hiding and D(B)DHI assumptions compare with the constructions based on DDH provided in [46, 75]. The performance evaluation is split into two parts. First, we discuss our HPRG constructions' efficiency and compare them with existing schemes. Later, we look at our OWFE constructions and compare their performance.

6.1 Hinting PRG: Comparing with DDH Construction

Here we discuss our HPRG constructions' efficiency and compare it with existing constructions. First, we provide an asymptotic comparison and give a more concrete comparison.

An asymptotic comparison. Let us start by recalling the HPRG construction based on DDH assumption, immediately derived from [75], which will serve as a focal comparison point to prior work.[1] In their construction,

[1]Koppula and Waters [75] constructed Hinting PRG based on CDH assumption. To provide a more fair comparison with our constructions, we simplify their construction by relying on DDH assumption, thereby removing the need for using hardcore predicates. This

the public parameters consist of $O(n^2)$ group elements, where n is the HPRG seed's length (/number of blocks). Here each block is associated with $O(n)$ group elements. Now for computing the output of the HPRG for any given block, the evaluator picks half of these associated group elements and sets the output as the product of the chosen group elements. More formally, the evaluator performs $O(n)$ group operations per block during HPRG evaluation.

Comparing that to our Φ-Hiding based HPRG construction described in Section 5.4, the public parameters consist of $2n$ (λ-bit) prime exponents along with the RSA modulus, extractor seed, group generator, and a hash key. For evaluating a single HPRG block, the evaluator needs to perform $O(n)$ exponentiations. However, using our dynamic programming technique described in Section 5.4.1, we can reduce the number of exponentiation operations needed per block to grow only logarithmically in n.

In our DDHI based construction described in Section 5.5, the public parameters consist of n group elements along with the group generator and a hash key. For evaluating a single HPRG block, the evaluator evaluates a degree-n polynomial symbolically and later on performs n exponentiation operations and n group operations. We summarize the asymptotic comparison of the Hinting PRG constructions in Table 6.1, where N and p are the recommended RSA modulus and elliptic curve field size for the target security λ.

---

leads to a more efficient setup phase and shorter public parameters and provides a more accurate baseline for comparing performance.

Metric	DDH [75]	Φ-Hiding (§5.4)	DDHI (§5.5)
Seed length n	$\log p + 2\lambda$	$\log N + 2\lambda$	$\log p + 2\lambda$
pp size	$O(n^2)$ group elements	$O(n\lambda)$ bits and $O(1)$ elements in $\mathbb{Z}_N^*$	$O(n)$ group elements
Time (Setup)	Sampling $O(n^2)$ group elements	Sampling $O(n)$ λ-bit primes	$O(n)$ exponentiations
Time (Eval)	$O(n^2)$ group operations	$O(n \log n)$ exponentiations	$O(n^2)$ exponentiations

Table 6.1: Asymptotic Performance Comparision of Various Hinting PRG Constructions.

Concrete performance evaluation. We performed the evaluations on a 2015 Macbook Pro with Dual Core 2.7 GHz Intel Core i5 CPU and 8GB DDR3 RAM. We evaluated the performance of DDH and DDHI based constructions using MCL Library [68] (written in C++) on NIST standardized elliptic curves P-192, P-224, P-256, P-384, and P-521, providing 96, 112, 128, 192, and 260-bit security, respectively. We evaluated our Φ-Hiding based construction using Flint Libary [65] written in C++ on 1024, 2048, 3072, 7680, and 15360 bit RSA modulus providing 80, 112, 128, 192 and 256-bit security, respectively.[2] The performance numbers are provided in Table 6.2. For Φ-hiding based HPRG, the numbers mentioned in the brackets indicate the estimated evaluation times (based on benchmarks) without the dynamic programming technique described in 5.4.1.

[2]Note that we proved the security of our schemes in an asymptotic sense. However, for experiments, we use NIST recommended RSA modulus for the sake of simplicity. The RSA modulus derived from applying a concrete analysis is slightly higher. However, as we mentioned in Footnote 1, the analysis could be improved further by using a tighter bound for $\Pi(r^i)$.

Now we present a few observations and interpretations of the above performance measures. We begin with the 112-bit security level as a focal point. The first thing that jumps out is that our constructions provide a rather dramatic trade-off between evaluation time versus parameter size compared to prior work. Please observe that the size of the public parameters for our Φ-Hiding and DDHI based constructions are ∼120x and ∼1900x shorter than the public parameter size provided by the baseline [75] DDH-based construction. The setup phase of our DDHI based construction is also ∼1450x faster than that of the DDH-based construction. On the flip side, in our Φ-Hiding based construction, the evaluation algorithm is ∼120x slower than that for the DDH-based construction. Moreover, our DDHI-based construction has a further ∼3x slowdown compared to our Φ-Hiding based construction. From the numbers, we can also see that our optimization for reusing computation across multiple RSA blocks is critical for the construction to be viable. The unoptimized version would take ∼180x more evaluation time than the optimized version.

Thus, the clear trade-off between the two constructions is between optimizing public parameters' size and reducing the HPRG evaluation algorithm's running time. We can also observe that as we move up security parameters, our DDHI based construction begins to dominate the Φ-hiding-based construction in all aspects. For 128 bit security, both the constructions' evaluation time is about even. In contrast, the parameter size of the DDHI based construction is considerably smaller. For larger security parameters DDHI based construction dominates completely. The reason for this is that due to the number field

sieve attacks, the recommended RSA modulus length (and thereby HPRG seed length n) increases super linearly with the target security level for the Φ-based construction. In comparison, the recommended field size (and thereby HPRG seed length n) will increase linearly for the elliptic curve DDH and DDHI based constructions. In the Hinting PRG context, this gives a double whammy to the Φ-hiding construction as an increase of n will increase both the number of blocks and the cost of group exponentiations required to compute each block. One can see that even for higher security parameters, the amortized average computation time per block for the Φ-hiding construction remains lower. However, the overall computation is higher due to the number of blocks needed.

Further reducing $|\mathsf{pp}|$ for Φ-hiding based construction. Observe that the public parameter size of the Φ-hiding-based construction is dominated by the $2n$ λ-bit primes $e_{j,b}$. To reduce the parameter size, one could employ the PRF-trick suggested by Hohenberger and Waters [70] in a different context. Their idea was to generate all the prime values $e_{j,b}$ from a PRF with a randomly chosen but publicly known seed. Using this approach, we can significantly reduce the parameter size. This means our Φ-hiding-based HPRG construction has the smallest public parameter length even for large values of targetted security levels. (The reason we could use a similar approach is that in each of the hybrids in our proofs, we only need to use exactly one of the $2n$ $e_{j,b}$ values for security, and the rest can be almost sampled arbitrarily.) However,

an important and subsequent trade-off is that since the $e_{j,b}$ values now would need to be computed on-the-fly during evaluation each time. This involves performing prime searches as part of the Eval algorithm and can lead to a further increase in evaluation time. Moreover, the proof of our construction would need to be adapted in a way similar to [70] to accommodate this change. One would additionally need to be extra careful in extending our hashing lemma (Theorem 4.2.1) to incorporate this change for the entire proof to work. Below in Table 6.3, we provide the performance metrics for such a modified construction.

CCA Security via Hinting PRGs. Although Hinting PRGs are an elegant cryptographic primitive, and therefore coming up with more efficient constructions is interesting in its own right. So far, the most prominent application of Hinting PRGs has been in upgrading any CPA-secure PKE/ABE scheme to be CCA-secure [75]. Here we thereby analyze how our Hinting PRG constructions affect the performance of the CPA to CCA-secure PKE/ABE transformation and briefly compare with other approaches for similar transformations.

Let us first briefly recall some important aspects of the CCA-secure PKE construction proposed by [75][3]. In their construction, the setup involves performing an HPRG Setup for sampling HPRG public parameters pp which

[3]In the original construction by [75], the setup algorithm of CCA-secure PKE samples $2n$ public-secret key pairs of the underlying CPA-secure PKE. Later in [74], it was suggested that one could instead sample only 2 public-secret key pairs of the underlying CPA-secure PKE. For an adequate analysis, here we always consider the optimized [75] construction as suggested in [74].

are included as part of the public encryption key. The encryptor runs the HPRG Eval algorithm once for each block during encryption. The ciphertext size also grows linearly with the seed length n. Now observe that the recommended HPRG seed length n is lowest for the DDH and DDHI-based constructions for any given security parameter. Additionally, DDHI-based construction leads to shorter public parameters. Thus, if one uses the DDHI-based HPRG construction proposed in this thesis, it leads to much smaller public encryption key and ciphertext sizes. The trade-off is higher encryption/decryption times. (On the other hand, if the goal is to minimize the size of the public encryption key, our Φ-Hiding construction with the aforementioned optimization technique could be used instead.) In conclusion, the trade-off between public parameter size and evaluation time in the HPRG constructions results in a trade-off between encryption key/ciphertext sizes and encryption/decryption times in the resultant CCA-secure construction.

We implemented the CPA to CCA-secure transformation for public-key encryption proposed by [75]. We started with the ElGamal encryption and transformed it into a CPA secure public-key encryption with randomness recovery property using the transformation proposed in [75]. We used ECDSA signatures for the transformation. We used the key-encapsulation mechanism (along with AES) to encrypt long ciphertexts. We used various hinting prg constructions and compared the efficiency of the final CCA secure construction.

The computational environment is set up as before. We performed the evaluations on a 2015 Macbook Pro with Dual Core 2.7 GHz Intel Core

i5 CPU and 8GB DDR3 RAM. We evaluated the performance of DDH and DDHI based constructions using MCL Library [68] (written in C++) on NIST standardized elliptic curves P-192, P-224, P-256, P-384, and P-521, providing 96, 112, 128, 192, and 260-bit security, respectively. We evaluated our Φ-Hiding based construction using Flint Libary [65] written in C++ on 1024, 2048, 3072, 7680, and 15360 bit RSA modulus providing 80, 112, 128, 192, and 256-bit security, respectively. We present the results in Table 6.4.

We observe that when our DDHI-based hinting prg is used for the transformation, the public key size, secret key size, ciphertext size, and setup time are the lowest. When DDH-based hinting proposed by [75] is used, the secret key size, ciphertext size, encryption, and decryption times are the lowest. Our Φ-hiding construction has a smaller public key size than the DDH-based construction. However, it did not perform well in terms of the other metrics due to large group sizes.

Comparing with [74] (CCA Security via KDM Security). In a follow-up work to [75], Kitagawa et al. [74] provided a similar transformation for achieving CCA security but by relying on Key Dependent Message secure SKE [12, 64, 69, 4] instead of Hinting PRGs. A natural question would be whether this approach would outperform the [75] construction after we plug in the HPRGs proposed in this thesis. It turns out that the [75] construction is asymptotically a lot more efficient than [74] in terms of key sizes, ciphertext sizes, setup, encryption, and decryption times. The reason is that

in [74] construction, most of the efficiency metrics (such as public key and ciphertext size, setup/encryption/decryption times) grow linearly with the key length ℓ of the underlying KDM-secure system. In most existing KDM-secure schemes [16, 19, 21], the key length ℓ is at least $O(\lambda^2)$ bits. Therefore, this approach leads to much worse (a quadratic slowdown) parameters than HPRG-based constructions. Thus, this further motivates the problem of improving the efficiency of Hinting PRGs.

6.2 OWF with Encryption: Comparing with DDH Construction

We now discuss our OWFE constructions' efficiency and compare them with existing constructions. First, we provide an asymptotic comparison and give a more concrete performance evaluation.

An asymptotic comparison. In the [46] construction, the public parameters consist of $O(n)$ group elements, where n is at least $\log p + 2\lambda$, and p is the group size. The function evaluation and decryption algorithm performs $O(n)$ group operations. The E_1 algorithm performs $O(n)$ exponentiations and outputs a ciphertext containing $O(n)$ group elements. The E_2 algorithm performs one exponentiation and outputs a key containing one group element.

Comparing that to our Φ-Hiding based OWFE construction described in Section 5.1, the public parameters consist of $2n$ (λ-bit) prime exponents along with the RSA modulus N, extractor seed, group generator, and a hash

key. The function evaluation and decryption algorithm performs $O(n)$ exponentiations with λ-bit exponents, where n is at least $\log N + 2\lambda$. Both E_1 and E_2 algorithms perform single exponentiation and output a ciphertext and key containing just one group element, respectively.

In our DDHI based construction described in Section 5.2, the setup phase performs $O(n)$ exponentiations and outputs public parameters containing n group elements, where n is at least $\log p + 2\lambda$, and p is the group size. The function evaluation and decryption algorithms evaluate a degree-n polynomial symbolically and perform n exponentiation operations and n group operations. The E_1 algorithm performs $O(n)$ exponentiations and outputs a ciphertext containing $O(n)$ group elements. The E_2 algorithm performs one exponentiation and outputs a key containing 1 group element. We also provide a more efficient OWFE construction Section 5.3 by relying on bilinear maps and prove it secure under DBDHI. It is similar to the DDHI based OWFE, except that the E_1 algorithm only performs $O(1)$ exponentiations, E_2 and decryption algorithms additionally perform a pairing operation, and ciphertext contains only one group element.

Concrete performance evaluation. The evaluations are performed in a computational environment similar to HPRG evaluation. Additionally, we evaluated the performance of DBDHI based OWFE using MCL Library [68] on BN-254, BN-381, BN-462 pairing-friendly elliptic curves [7] (providing 100, 128, 140-bit security after the recent tower number field sieve attacks [73, 80,

43]).

It turns out that the baseline DDH based OWFE offers the shortest setup, evaluation, and decryption times. In comparison, the Φ-hiding-based OWFE outperforms in terms of E_1 time and ciphertext size. Due to smaller group size (and thereby smaller n), DBDHI based OWFE leads to the shortest E_1 time and ciphertext size. Lastly, for the shortest E_2 time and key size, both the DDH and DDHI based constructions are equally useful. The concrete performance numbers are provided in Table 6.5.

Note that even though both DDHI and DBDHI based OWFE schemes have the same one-way function, DDHI based scheme has a faster evaluation time. In fact, the DDHI based construction is more efficient than DBDHI construction in all aspects other than E_1 time and ciphertext size. This is because the recommended group size of pairing-based elliptic curves grows super linearly in the security parameter due to the number field sieve attacks. Furthermore, the function evaluation and decryption procedures of Φ-hiding based scheme perform $O(n)$ exponentiations, compared to $O(n)$ group operations performed by other schemes. As a result, the Φ-hiding-based scheme has the slowest function evaluation and decryption procedures.

Deterministic Encryption from OWFE. A fascinating application of OWFE is of deterministic encryption as shown by [44]. In the deterministic encryption scheme of [44], the setup phase invokes the OWFE setup phase once and E_1 algorithm $O(\ell)$ times, where ℓ is proportional to the length of the mes-

sage being encrypted. The encryption key includes OWFE public parameters and $O(\ell)$ OWFE ciphertexts. The encryption algorithm invokes OWFE f algorithm once and OWFE D algorithm $O(\ell)$ times. The decryption algorithm invokes OWFE E_2 algorithm $O(\ell)$ times. Consequently, our DBDHI based OWFE leads to a deterministic encryption scheme with much smaller public parameters and setup time. Concretely, at 128-bit security, the setup phase and public parameters of our DBDHI based deterministic encryption scheme for 128-bit messages are more than 200x faster and 240x shorter, respectively, than the baseline DDH based deterministic encryption described in [44].

Metric	Security	DDH [75]	Φ-Hiding (§5.4)	DDHI (§5.5)
Seed Length n	80/96	384	1184	384
	112	448	2272	448
	128	512	3328	512
	192	768	8064	768
	256/260	1042	15872	1042
pp size	80/96	28.3 MB	0.07 MB	0.018 MB
	112	25 MB	0.19 MB	0.024 MB
	128	67.16 MB	0.32 MB	0.032 MB
	192	226.64 MB	1.16 MB	0.074 MB
	256/260	537.12 MB	3.05 MB	0.13 MB
Time (Setup)	80/96	7.57s	1.32s	0.0285s
	112	83.99s	6.22s	0.058s
	128	15.95s	11.86s	0.080s
	192	102.68s	98.74s	0.370s
	256/260	431.78s	443.84s	1.598s
Time (Eval)	80/96	0.042s	1.07s (112.5s w/o opt.)	14.24s
	112	0.086s	11.20s (2164.2s w/o opt.)	30.13s
	128	0.11s	40.79s (3.32 hrs w/o opt.)	46.77s
	192	0.48s	719.479s (5.46 days w/o opt.)	284.83s
	256/260	2.145s	5975.34s (84.27 days w/o opt.)	1515.68s
Eval time per block	80/96	0.109ms	0.904ms (0.11s w/o opt.)	37.09ms
	112	0.192ms	4.93ms (1.05s w/o opt.)	67.26ms
	128	0.215ms	12.26ms (3.88s w/o opt.)	91.35ms
	192	0.625ms	89.22ms (61.45s w/o opt.)	349.98ms
	256/260	2.06ms	376.47ms (473.36s w/o opt.)	1454.59ms

Table 6.2: Performance comparison of various Hinting PRG constructions. The time taken by Eval algorithm without the optimization described in Algorithm 1 is presented in braces.

Metric	80	112	128	192	256
Seed Length n	1184	2272	3328	8064	15872
pp size	0.77 KB	1.54 KB	2.30 KB	5.76 KB	11.52 KB
Time (Setup)	0.0079s	0.067s	0.247s	6.73s	82.61s
Time (Eval)	2.70s	17.60s	52.61s	813.28s	6342.10s

Table 6.3: Performance metrics of the Φ-Hiding based HPRG optimized for small pp size

Metric	Security	DDH [75]	Φ-Hiding (§5.4)	DDHI (§5.5)
pk size	80/96	28.4 MB	0.36 MB	0.113 MB
	112	45.09 MB	0.86 MB	0.16 MB
	128	67.28 MB	1.44 MB	0.20 MB
	192	226.88 MB	5.22 MB	0.46 MB
	256/260	566.39 MB	13.87 MB	0.85 MB
sk size	80/96	9.2 KB	28.41 KB	9.2 KB
	112	12.54 KB	63.62 KB	12.54 KB
	128	16.38 KB	106.50 KB	16.38 KB
	192	46.86 KB	387.07 KB	46.86 KB
	256/260	67.86 KB	1033.66 KB	67.86 KB
ct size	80/96	0.10 MB	0.32 MB	0.10 MB
	112	0.15 MB	0.73 MB	0.15 MB
	128	0.19 MB	1.22 MB	0.19 MB
	192	0.43 MB	4.45 MB	0.43 MB
	256/260	0.78 MB	11.86 MB	0.78 MB
Time (Setup)	80/96	7.64s	1.51s	0.099s
	112	84.11s	6.83s	0.176s
	128	16.11s	12.89s	0.236s
	192	103.42s	106.57s	1.11s
	256/260	434.99s	492.95s	4.81s
Time (Enc)	80/96	0.16s	1.44s	14.36s
	112	0.23s	12.41s	30.37s
	128	0.42s	42.83s	47.08s
	192	1.96s	750.05s	286.31s
	256/260	8.51s	6073.34s	1522.05s
Time (Dec)	80/96	0.163s	1.45s	14.363s
	112	0.234s	12.43s	30.374s
	128	0.424s	42.86s	47.084s
	192	1.966s	750.14s	286.316s
	256/260	8.518s	6073.53s	1522.058s

Table 6.4: Performance metrics the final CCA secure scheme upon applying the Koppula and Waters [75] on the ElGamal encryption scheme along with various Hinting PRG constructions discussed in this thesis and prior works

Metric	Security	DDH [46]	Φ-Hiding (§5.1)	DDHI (§5.2)	DBDHI (§5.3)
pp Size	80/96/BN254	36.8 KB	71.8 KB	18.4 KB	28.8 KB
	112	50.2 KB	192.4 KB	25.1 KB	-
	128/BN381	65.4 KB	321.8 KB	32.8 KB	60.8 KB
	140/BN462	-	-	-	85.7 KB
	192	147.4 KB	1167 KB	73.8 KB	-
	256	262.2 KB	3059 KB	131.4 KB	-
ct Size	80/96/BN254	36.8 KB	128 Bytes	18.4 KB	128 Bytes
	112	50.2 KB	256 Bytes	25.1 KB	-
	128/BN381	65.4 KB	384 Bytes	32.8 KB	192 Bytes
	140/BN462	-	-	-	232 Bytes
	192	147.4 KB	960 Bytes	73.8 KB	-
	256	262.2 KB	1920 Bytes	131.4 KB	-
Key Size	80/96/BN254	48 Bytes	128 Bytes	48 Bytes	762 Bytes
	112	56 Bytes	256 Bytes	56 Bytes	-
	128/BN381	64 Bytes	384 Bytes	64 Bytes	1146 Bytes
	140/BN462	-	-	-	1186 Bytes
	192	96 Bytes	960 Bytes	96 Bytes	-
	256	128 Bytes	1920 Bytes	128 Bytes	-
Time (Setup)	80/96/BN254	0.0096s	1.40s	0.026s	0.0435s
	112	0.093s	6.69s	0.052s	-
	128/BN381	0.016s	12.43s	0.070s	0.158s
	140/BN462	-	-	-	0.493s
	192	0.065s	101.38s	0.307s	-
	256	0.203s	475.55s	1.326s	-
Time (f)	80/96/BN254	0.0001s	0.11s	0.037s	0.059s
	112	0.0002s	1.06s	0.068s	-
	128/BN381	0.0002s	3.67s	0.090s	0.19s
	140/BN462	-	-	-	0.54s
	192	0.0006s	59.14s	0.353s	-
	256	0.0020s	473.36s	1.41s	-
Time (E_1)	80/96/BN254	49.1ms	0.69ms	29.44ms	0.188ms
	112	100.87ms	3.10ms	56.80ms	-
	128/BN381	134.90ms	9.40ms	76.40ms	0.45ms
	140/BN462	-	-	-	1.435ms
	192	600.84ms	106.57ms	326.49ms	-
	256	2590.14ms	601.5ms	1357.93ms	-
Time (E_2)	80/96/BN254	0.067ms	0.40ms	0.066ms	0.68ms
	112	0.12ms	2.80ms	0.11ms	-
	128/BN381	0.14ms	8.38ms	0.136ms	1.79ms
	140/BN462	-	-	-	4.52ms
	192	0.40ms	99.50ms	0.40ms	-
	256	1.26ms	600.03ms	1.29ms	-
Time (D)	80/96/BN254	0.0001s	0.109s	0.036s	0.059s
	112	0.0003s	1.09s	0.067s	-
	128/BN381	0.0003s	3.57s	0.090s	0.19s
	140/BN462	-	-	-	0.54s
	192	0.00083s	58.96s	0.355s	-
	256	0.00286s	466.84s	1.41s	-

Table 6.5: Concrete performance evaluation of various OWFE constructions

Chapter 7

Registration-Based Encryption

In the previous chapters, we defined one-way function with encryption, described its importance, presented many constructions, and compared their performance. In the next few chapters, we present an application of a one-way function with encryption called registration-based encryption. We first describe the key escrow problem in identity-based encryption, and more generally, functional encryption. We then review a solution to the key-escrow problem proposed by Garg et al. [47]. Garg et al. defined a novel encryption mechanism called registration-based encryption to solve the key-escrow problem. They further constructed it from the one-way function with encryption. We analyze the solution and observe that their proposal does not entirely solve the key-escrow problem. We further add a property called "verifiability" to the notion of registration-based encryption and define "verifiable registration-based encryption" that solves the key-escrow problem. We further construct this encryption mechanism based on some of the ideas used by Garg et al. [47, 48] along with our idea of "snapshotting".

7.1 Key Escrow Problem

Public-key encryption (PKE) [34, 88, 56] has remained a cornerstone in modern-day cryptography and has been one of the most widely used and studied cryptographic primitive. Traditionally, a public-key encryption system enables a one-to-one private communication channel between any two users over a public broadcast network as long as it is possible to disambiguate any user's public key information honestly. Over the last few decades, the cryptographic community has made significant research efforts in re-envisioning the original goals of public-key encryption, pushing towards more expressiveness from such systems. This effort has lead to introduction of encryption systems with better functionalities such as Identity-Based Encryption (IBE) [95, 32, 15], Attribute-Based Encryption (ABE) [90, 62], and most notably Functional Encryption (FE) [91, 17] which is meant to encapsulate both IBE and ABE functionalities.

Very briefly, in FE systems, there is a trusted authority that sets up the system by sampling public parameters pp along with a master secret key msk. The public parameters pp can be used by any party to encrypt a message m of its choice, while the master key msk enables the generation of certain private decryption keys sk_f for any function f in the associated function class. The most useful aspect of such systems is that decryption now leads to users either conditionally learning the full message (as in IBE/ABE where the condition is specified at encryption time) or learning some partial information about the message such as $f(m)$ (as in general FE). The security of all such systems

guarantees that no computationally bounded adversary should be able to learn anything other than what can be uncovered using the private decryption keys in its possession.

Notably, in all such highly expressive systems, the master key msk must never be compromised. Given the master key, any adversary can arbitrarily sample private decryption keys to learn desired messages. Thus, an unfortunate consequence of adding such powerful functionalities to public-key cryptosystems is introducing a central trusted authority (or key generator). The authority is responsible for sampling the public parameters, distributing the private decryption keys to authorized users, and securely storing the master secret key. This could be very worrisome for many applications since the authority must be fully trustworthy. Otherwise, it would turn out to be a single point of failure. While it would be quite reasonable to trust the central authority, it so happens that even an honest-but-curious key generator can cause great havoc in such an environment. Specifically, any honest-but-curious key generator can arbitrarily decrypt ciphertexts intended for specific recipients since it has the master key. Furthermore, it could perform such an attack in an undetectable way. This problem is widely regarded as the "key-escrow" problem.

7.2 Prior Approaches to the Key Escrow Problem

The problem of reducing the amount of trust put behind private key generators (PKGs) in an IBE environment has been well studied. Boneh and

Franklin [15] made initial attempts. They suggested using multiple PKGs, instead of just one, to decentralize the PKG, thereby preventing PKGs from being a single target of corruption. This idea was further explored in many subsequent works ([28, 76, 85, 71] to name a few). Later on, Goyal [60] considered an alternate approach in which he studied the notion of accountable authority IBE. In such a system, the goal is to make the PKG accountable by requiring that any malicious activity by the PKG (which could involve distributing decryption keys to unauthorized users) can be traced back to the cheating authority. Goyal [60] was able to show that Gentry's scheme [52] was already an accountable IBE scheme and provided a different construction based on the DBDH assumption. In follow-up work, [61] gave a construction of black-box accountable authority IBE from DBDH Assumption where tracing guarantee was much stronger.

Al-Riyami and Paterson [2] put forward the notion of "Certificateless" Public Key Cryptography in which the key generation process is split between a central authority and the user. The authority first generates a key pair, where the private key is now the user's partial private key. The remainder of the key is a random value generated by the user. It is never revealed to anyone, not even the authority. A crucial bottleneck in such systems is that an encryptor needs to obtain additional information about the user from the authority before encrypting a message. Building on this, [29] proposed a similar setting and gave a construction based on [15] scheme.

Another line of work has been centered around trying to solve the

key escrow problem by relying on anonymous IBE [14]. This approach was originally studied in [31], where the intuition was that since in an anonymous IBE system ciphertexts hide the recipient identity, therefore key generation authority's ability to decrypt the ciphertext is adversely affected whenever there is some min-entropy in the space of identities (e.g., biometric identities). A similar approach was later taken in [100] as well.

7.3 Registration-Based Encryption

While many previous works ([15, 28, 2, 29, 60, 85, 31, 71] to name a few) have suggested different approaches to solving the key-escrow problem, none of these solutions were able to resolve the key-escrow problem completely. Very recently, in a beautiful work by Garg, Hajiabadi, Mahmoody, and Rahimi [47], a novel approach for handling key-escrow was proposed. The central motivation of that work was to remove private key generators' requirements from IBE systems completely. To that end, they introduced the notion of Registration-Based Encryption (RBE). In an RBE system, each user samples its own public-secret key pair, and the private key generator is replaced with a public key accumulator. Every user registers their public key and identity information with the key accumulator. The key accumulator's job is compressing all these user identity-key pairs into a short public commitment with efficiently computable openings. Here the commitment is set as the public parameters of the RBE system, and the user-specific openings are used as supplementary key information during decryption. Ideally, one would expect the registration

process to be time-unrestricted. Users must be allowed to register at arbitrary time intervals. However, this would imply that public parameters will get updated after every registration, leading to every registered user requesting fresh supplementary key information after another user registers. Thus, to make the notion more attractive, [47] required the following efficiency properties from an RBE system — (1) public parameters must be short, i.e., $|\mathsf{pp}| = \mathsf{poly}(\lambda, \log n)$ where λ is the security parameter, and n is the number of users registered so far, (2) the registration process, as well as the supplementary key generation process, must be efficient, i.e., must run in time $\mathsf{poly}(\lambda, \log n)$, (3) Number of times any user needs to request a fresh supplementary key from the accumulator (over the lifetime of the system) is also $\mathsf{poly}(\lambda, \log n)$. In short, an RBE system is meant to be a public key accumulation service that provides efficient and adaptive user registration while avoiding the problems associated with a private key generator.

In a sequence of two works [47, 48], efficient construction of RBE systems were provided from a wide variety of assumptions (such as CDH, Factoring, LWE, iO). Specifically, [47] gave an efficient construction from indistinguishability obfuscation (iO) [5, 45], and a weakly efficient construction from hash garbling scheme [47]. In a follow-up work by Garg et al. [48], a fully efficient RBE construction from hash garbling was provided. At first glance, it seems like efficient constructions for RBE systems fill the gap between regular PKE systems (which do not suffer from key-escrow but also do not provide any extra functionality) and IBE systems (which permit a simpler identity-based

encryption paradigm but suffers from key-escrow). However, it turns out there is still a significant gap due to which even RBE systems potentially could be surprisingly compromised due to a corrupt key accumulator. To better understand the gap, let us look back at the excerpt from Rogaway's essay [89] which was one of the prompts behind the initial work on RBE in [47]:

> *"[. . .] But this convenience is enabled by a radical change in the trust model: Bob's secret key is no longer self-selected. A trusted authority issues it. That authority knows everyone's secret key in the system. IBE embeds key-escrow, indeed a form of key-escrow where a single entity implicitly holds all secret keys, even ones that have not yet been issued. [. . .] "*

Now an RBE system solves the problem of self-selection of Bob's (or any user's) secret key faced in IBE systems. That is, during honest registration, every user samples its own public-secret key pair. However, the *embedding key-escrow problem* is still not directly prevented by the RBE abstraction. This is because a dishonest key accumulator could potentially add either certain trapdoors, or secretly register multiple keys for already registered users, or register any key for currently unregistered users. There could be many such scenarios in which a key accumulator's malicious behavior permits decryptability of ciphertexts intended towards arbitrary users by the key accumulator depending upon its adversarial strategy. Although such attack scenarios were not explicitly studied in the prior works [47, 48], a beneficial by-product of

the approaches taken in those works was that the user registration process (and all the computations performed by the key accumulator) was completely deterministic. It thus leads to a straightforward and elegant methodology for avoiding the embedding key-escrow problem by providing full public auditability. Basically, any user (or even a non-participating third party) could audit the key accumulator and verify honest behavior by rebuilding the RBE public parameters and comparing that with the accumulated public parameters. As honestly generated public parameters do not have any trapdoors or faulty keys embedded by construction, public auditability solves the embedding key-escrow problem.

Although the above deterministic reconstructability feature of the RBE systems serves as a possible solution to the embedding key-escrow problem, this is not efficient. Concretely, if any new (or even already registered) user wants to verify that the key accumulation has been done honestly, then that particular user needs to obtain a $O(n)$ (linear-sized) proof as well as spend $O(n)$ (linear amount of) time for verification, where n is the number of users registered until that point. In this thesis, we study whether we can build RBE systems where such verifications could be sped up. Specifically, we ask the following:

> *Do there exist efficient registration-based Encryption schemes in which any user can obtain short proofs of unique registration as well as short proofs of non-registration? Can such proof mechanisms be*

useful for speeding up the auditability process? Is it even possible to provide all such guarantees with only a $\mathsf{poly}(\lambda, \log n)$ *cost incurred in the size of proof and running time of provers/verifiers?*

We answer the above questions in affirmative by introducing a notion of efficient verifiability for RBE systems and providing an instantiation from hash garbling schemes [47] thereby giving constructions based on standard assumptions (such as CDH, Factoring, LWE). Concretely, our contributions are described below.

7.4 Overview of Our Results

In this thesis, we introduce a new notion for key accumulation, which we call Verifiable registration-based Encryption (VRBE). Briefly, a VRBE system is simply a standard RBE system in which the key accumulator can also provide proofs of correct (and unique) registration for every registered user as well as proofs of non-registration for any yet unregistered identity. We give new constructions for VRBE from hash garbling schemes that provide succinct proofs of registration and non-registration. Here, the key accumulator can efficiently carry out the registration and proof generation processes. Our proof systems also naturally allow a much more efficient audit process that any non-participating third party can perform. A by-product of our approach is that we provide a more efficient RBE construction than that provided in the most recent work of Garg et al. [48], wherein the size of ciphertexts in our

construction is significantly smaller.[1] And, lastly, we briefly discuss how the notion of VRBE can also be naturally extended to a wider range of access and trust structures, wherein the keys accumulated are no more associated with a PKE system but for even more expressive encryption systems. Such systems might be practically more interesting in the future.

Next, we provide a detailed overview of our approach and describe the technical ideas. We start by recalling the notion of RBE as it appears in prior works. We then discuss our proposed notions of efficient verifiability for such systems. Since the starting point of our construction is the RBE scheme proposed in [48], we first recall the main ideas and high-level structure of their approach. We later describe our construction and show how to provide succinct proofs of registration and non-registration for any user identity, thereby adding verifiability to the system.

7.5 RBE Abstraction

In an RBE system, there is a dedicated party which we call the key accumulator. A key accumulator runs the registration procedure indefinitely[2], where any user could make one of two types of queries — (1) registration

[1]Looking ahead, our efficiency gain is because our construction takes a one-shot (single-step) approach whereas [48] takes a two-step approach. Here the outcome of a two-step approach is that the ciphertext consists of two layers of cascaded garbled circuits. In contrast, our solution consists of a single sequence of garbled circuits.

[2]It could run it sporadically as well, where it simply records all new registrations made in a certain time window and later registers them all at once. For simplicity, here we consider the key accumulator is always online.

query, where a new user sends in its identity and public key pair $(\mathsf{id}, \mathsf{pk})$ for registration, (2) update query, where an already registered user requests for supplementary key material u which is used for decryption. (The supplementary key material is usually referred to as the update information.) The key accumulator maintains the public parameters pp along with some auxiliary information aux throughout its execution. After each registration query of the form $(\mathsf{id}, \mathsf{pk})$, it updates the parameters to pp' and aux' to reflect addition. It then sends back the associated key material u to the corresponding user. For each update query made by a user with identity id, the accumulator extracts an update u from the auxiliary information aux and sends it over to the user. The encryption and decryption procedures are defined analogous to the IBE counterparts, except during decryption, a user needs a piece of appropriate update information u to complete the operation.

At a high level, the correctness requirement states that any honestly registered user with identity id and key pair $(\mathsf{pk}, \mathsf{sk})$ must be able to decrypt a ciphertext encrypted for the identity id under public parameters pp (which could have been updated after id was registered) using its own secret key sk as long as it also gets an update information u corresponding to pp from the accumulator. For efficiency, it is important that the size of public parameters pp, size of update information u, and the number of updates required by any user throughout its lifetime grow at most poly-logarithmically with the number of registered users n. Additionally, the registration process and update generation should run in time $\mathsf{poly}(\lambda, \log n)$. Lastly, for security, it is essential

that a ciphertext encrypting a message m for an identity id under parameters pp should hide the message as long as either id was not registered by the time pp was computed, or the key pair registered with identity id was honestly sampled and the corresponding secret key is unknown to an attacker.

7.5.1 Inadequacies of RBE and Workarounds

As we discussed before, the above abstraction still suffers from the embedding key-escrow problem. Specifically, the RBE system does not provide any abstraction for efficiently verifying whether a dishonest key accumulator — (1) secretly registers a public-secret key pair for any yet unregistered identity, (2) or while registering any new user (or even at any later point in time), also introduces a trapdoor (or register multiple keys for the same identity) that enables unauthorized decryption. For this specific reason, we study the possibility of efficient verifiability for RBE systems. Concretely, we consider two orthogonal notions of verifiability for an RBE scheme — *pre-registration* and *post-registration* proofs. Intuitively, the goal of pre-registration verifiability is to provide a short proof π validating that a given id has not yet been registered as per public parameters pp and any ciphertext ct encrypted towards such an identity id will completely hide the plaintext even if all other secret keys are leaked. Similarly, the intuition behind post-registration verifiability is to provide a short proof π of *unique* accumulation, where the proof π guarantees that the key accumulator must not have added a trapdoor (or doubly registered) during a possibly dishonest registration which allows decryption

of ciphertexts intended for that particular user. (Looking ahead, our formal definitions of pre/post-registration verifiability are stated in a much stronger way where we allow an adversary to completely control the key accumulator and still require the soundness/message-hiding property to hold.)

7.6 Defining Verifiable RBE

Formally, a verifiable RBE system is just like a regular RBE system with four additional (deterministic) algorithms — $\mathsf{PreProve}$, $\mathsf{PostProve}$, $\mathsf{PreVerify}$, and $\mathsf{PostVerify}$. The pre-registration prover takes as input a common reference string crs, public parameters pp, and a target identity id for which a proof π of pre-registration is provided. On the other hand, the post-registration prover also takes a target public key pk as input. Both these provers are given random-access to the auxiliary information for time-efficient computation. Informally, the completeness of these proof systems states that the pre/post-registration verifier should always accept honestly generated proofs. Furthermore, for soundness, the requirement is that if the pre-registration verifier accepts a proof π for an identity id w.r.t. parameters pp, then ciphertexts encrypted towards id must hide the message completely from a malicious key accumulator which computes the parameters pp and proof π. Similarly, for post-registration soundness, the property states that if the verifier accepts a proof π for identity-key pair $(\mathsf{id}, \mathsf{pk})$ w.r.t. parameters pp, then ciphertexts encrypted towards id must hide the message completely from a malicious key accumulator as long as the accumulator does not possess the secret key sk

associated with the public key pk.

Stronger correctness guarantees. In addition to the above-stated properties, we also define a solid form of extractable correctness property for our post-registration proof. Concretely, the extractable correctness property states that there exists a deterministic update extraction algorithm such that if there exists an accepting post-registration proof π for identity-key pair $(\mathsf{id}, \mathsf{pk})$ w.r.t. parameters pp, then the extraction algorithm computes update information u from the proof π itself such that using update u, anybody could decrypt ciphertexts encrypted for the identity id. Intuitively, extractable correctness states that completeness would still hold even for maliciously generated proofs. Our definitions are formally introduced later in Chapter 8.

A simple paradigm for efficient auditability. Looking back at our verifiability properties, one could interpret them as follows. The pre/post-registration proofs help ensure that a key accumulator behaves honestly at least *locally*. The idea behind a more global verification process is to perform the pre/post-registration verification on a randomly chosen (small) subset of users similar to what is done in probabilistically-checkable proof (PCP) literature. Specifically, suppose that a party claims that it has accumulated public parameters pp with the list of registered users R and non-registered users S. Any third party can efficiently audit the registration process by proceeding as follows — it samples a random subset of users in R and S, it requests post-registration

and pre-registration proofs for users in those subsets, respectively. If all the proofs are valid, the auditor approves that registration was done honestly. Note that depending upon the desired soundness threshold, the auditor can appropriately set the size of the subsets it samples. Thus, such randomized auditing would be more efficient than rebuilding the entire registration logs for most parameter regimes.

7.7 Reviewing Prior RBE Systems [47, 48]

Before outlining our approach, we quickly recall the high-level structure used in prior works [47, 48] since our construction uses similar building blocks. Let us first look at the weakly efficient RBE construction provided in [47] since it lays the major groundwork for the follow-up works (including ours). Their construction ideas can be summarized as follows at a high level. The key accumulator stores all the registered identity-key pairs $\{(\mathsf{id}_i, \mathsf{pk}_i)\}_{i\in[n]}$ using a shortlist of Merkle tree $\mathsf{Tree}_1, \mathsf{Tree}_2, \ldots, \mathsf{Tree}_k$, where every tree Tree_i is at least twice as large as Tree_{i+1}. Here the leaves of each tree encode one of the registered identity-key pairs $(\mathsf{id}, \mathsf{pk})$, while the internal nodes are like standard Merkle tree nodes (which is that they encode the hash of its children) except each node also stores the largest identity registered in its left sub-tree as well. In other words, each tree Tree_i is a binary search Merkle tree, with all the leaves are lexicographically sorted as per the identities. Consequently, any registered identity's public key can be obtained efficiently via a binary search. Each Merkle tree's root values serve as a short commitment to the entire registry

tree. To encrypt a message m to identity id, the encryptor needs to search the Merkle trees to obtain id's public key pk. However, the public parameters only contain the root node, not the entire tree. To overcome this issue, the [47] construction uses the ideas developed in a long line of works [30, 38, 42, 21, 40] of deferring the binary search to the decryption side by sending a set of garbled circuits as part of the ciphertext. Basically, for decryption, a user needs to obtain an opening (i.e., the path of nodes from root to leaf) in one of the Merkle trees to its registered key. This corresponds to the supplementary key material. Now, what makes the registration process only weakly efficient is that in order to register an identity-key pair $(\mathsf{id}, \mathsf{pk})$, the key accumulator creates a new tree consisting of only node $(\mathsf{id}, \mathsf{pk})$, and then merges all Merkle trees of equal size. This helps in keeping the size of the public parameters short. Since the leaves of the Merkle trees have to be sorted, the tree merging process is quite inefficient, which results in only a weakly efficient system.

In the follow-up work [48], the authors observed that the weakly efficient RBE construction described above is fully efficient if the identities being registered are already coming in sorted order. They call RBE schemes with these restrictions as Timed-RBE (T-RBE). Starting with this observation, they suggest a powerful two-step approach for building an efficient RBE system without this restriction, i.e., they provide a nice bootstrapping construction from T-RBE to general (non-timed) RBE with full efficiency. In their construction, the key accumulator associates every identity id with a timestamp t_{id} as well, where t_{id} is an internal counter incrementally maintained by the

accumulator. The idea is that since timestamps t_{id} will be accumulated in sorted order, for storing the association between the timestamp t_{id} and public key $\mathsf{pk}_{\mathsf{id}}$ one could use the T-RBE scheme. And, for storing the association between identity id and timestamp t_{id}, the accumulator maintains a *balanced* Merkle tree $\mathsf{TimeTree}$. The leaves of $\mathsf{TimeTree}$ encode the identity-timestamp pairs $(\mathsf{id}, t_{\mathsf{id}})$ for all registered users and are sorted as per the identities. The most crucial aspect of $\mathsf{TimeTree}$ is that it is balanced (for instance, they use a red-black tree). Let us look into more details about how such an additional balanced Merkle tree is useful in improving efficiency.

The key accumulator stores all the registered identity-timestamp pairs $\{(\mathsf{id}_i, i)\}_{i\in[n]}$ using a balanced Merkle $\mathsf{TimeTree}$, and stores the timestamp-key association using a small list of (standard) Merkle trees $\{\mathsf{Tree}_j\}_j$ as in [47]. The public parameters consist of multiple versions of the root node of the $\mathsf{TimeTree}$ along with the root nodes for $\{\mathsf{Tree}_j\}_j$. (Specifically, the public parameters store the root node and depth information of $\mathsf{TimeTree}$ for all timestamps whenever the underlying T-RBE Merkle tree was updated.) To register an $(\mathsf{id}, \mathsf{pk})$ pair, the key accumulator inserts the identity-timestamp pair $(\mathsf{id}, t_{\mathsf{id}})$ into $\mathsf{TimeTree}$, and timestamp-key $(t_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$ to the sequence of T-RBE trees. The most important aspect of the construction is that if the T-RBE trees storing timestamp-key associations are merged, then the versions of root nodes being stored in the public parameters are also updated. Next, let us look at how encryption and decryption are performed since the registration algorithm's efficiency follows almost immediately.

While encrypting message m for the identity id, the encryptor now provides two levels of garbled circuit sequences. The first level of garbled circuit sequence is used to find the timestamp t_{id} associated with id. In the next level, one uses the T-RBE garbled circuit sequence to encrypt m under the corresponding public key pk. They employ the same approach of deferring binary search to decryption for building both levels of garbled circuit sequences. The supplementary key material (or update) consists of two distinct paths. The first path is w.r.t. the $\mathsf{TimeTree}$ and the second path is as per the T-RBE system, which is w.r.t. one of the Merkle trees in $\{\mathsf{Tree}_j\}_j$. The most important component of this extended construction is that in order to tightly bound the number of updates (for any user identity id), the first portion of key material/update u (required for evaluating the first level of garbled circuits) are only issued whenever the first T-RBE Merkle tree in which identity id was registered gets merged. It turns out that executing the above idea formally leads to an efficient RBE scheme.

7.8 Our Verifiable RBE Solution

Our construction's starting point is the [48] RBE scheme described above. As a first step, we start by simplifying their construction and present a one-shot (single-step) approach to building efficient RBE systems. Later we describe how to make the simplified system verifiable, both in pre-registration and post-registration settings, without making any additional assumptions. Lastly, we provide some comparisons and discuss potential generic methods

for making existing RBE schemes verifiable.

Although the basic principles behind our simplified construction and the one provided in [48] are quite similar, there are significant structural differences in both approaches. Therefore, we provide a direct outline of our construction instead of going through the [48] construction and explaining the differences. Later on, we briefly compare our construction with theirs. Below we sketch the main ideas behind our construction. The actual construction is a little more complicated but follows naturally from the following outline. A detailed description appears later in Chapter 9.

In our construction, the key accumulator maintains a single *balanced* Merkle tree, which directly stores the mapping between identities and their respective public keys. Concretely, the key accumulator stores a balanced Merkle tree, which we call $\mathsf{EncTree}$. It consists of two types of nodes — leaf and intermediate. Like existing works, a leaf node stores an identity-key pair $(\mathsf{id}, \mathsf{pk})$ for every registered identity. In contrast, an intermediate node stores a tuple of the form $(h_{\mathsf{left}}, \mathsf{id}, h_{\mathsf{right}})$, where h_{left} and h_{right} are hash values of its left and right child (respectively) and id is the largest identity in its left sub-tree. Since $\mathsf{EncTree}$ is balanced and the nodes are ordered as per the registered identities, given an identity id, the key accumulator could efficiently search its associated public key (if id has been registered) and efficiently insert a new identity-key pair. The key accumulator stores $\mathsf{EncTree}$ as auxiliary information aux and publishes root value rt and depth d of the tree as part of public parameters pp. The registration algorithm inserts given identity-key

pair $(\mathsf{id}, \mathsf{pk})$ as a leaf in the $\mathsf{EncTree}$, balances the tree, and updates the hash values stored in all the ancestors of the newly inserted leaf. The registration algorithm then updates the public parameters pp to store the root value and depth of the updated $\mathsf{EncTree}$.

7.8.1 Encryption and Decryption

The encryption and decryption procedures follow the aforementioned 'deferred binary search' approach in which the ciphertext for the identity id contains a sequence of d garbled circuits, which work as follows. Given a path (a sequence of nodes from root to a leaf) in $\mathsf{EncTree}$ as input, the sequence of garbled circuits jointly checks that the path is well-formed. The leaf node encodes the identity id and outputs a PKE ciphertext under the public key encoded in the leaf node. Individually, the i^{th} garbled circuit performs the local well-formedness check on the path and outputs the garbled input for $(i+1)^{th}$ garbled circuit. For decryption, the decryptor needs to obtain a valid path u from the accumulator. The accumulator can efficiently generate u by performing a binary search on the $\mathsf{EncTree}$. Given a well-formed path, the decryptor can sequentially evaluate the garbled circuits and eventually obtain a PKE ciphertext, which it decrypts using its secret key.

7.8.2 How to Get the Desired Efficiency? The Snapshotting Trick

The above scheme is highly inefficient since updates must be issued each time a new user joins. At a very high level, we visualize our approach

to improve efficiency as that of storing multiple *'snapshots'* of the registration process, where an older snapshot is deleted only after new user registration leads to a new snapshot that as many users use as those using the older snapshot.[3] The intuition is to split the registered user space into disjoint groups of sizes — $1, 2, 4, \ldots, 2^{\lambda}$. For each group size, the public parameters will consist of *at most* one snapshot, which consists of the root node and depth information of (a possibly older version of) the balanced Merkle tree $\mathsf{EncTree}$.

Concretely, the public parameters look like $\{(j_1, \mathsf{snapshot}_{j_1}), \ldots, (j_\ell, \mathsf{snapshot}_{j_\ell})\}$ where every $j_i \in \{1, 2, \ldots, 2^{\lambda}\}$, and $j_i > j_{i+1}$, and $\mathsf{snapshot}_{j_i}$ consists of a root node and tree depth. These public parameters are interpreted as follows:

(1) the first j_1 users who registered refer to the $\mathsf{EncTree}$ corresponding to $\mathsf{snapshot}_{j_1}$ for decryption/obtaining update information;

(2) next j_2 users who registered refer to the $\mathsf{EncTree}$ corresponding to $\mathsf{snapshot}_{j_2}$;

$\vdots$

(ℓ) and similarly the last j_ℓ users to register refer to the $\mathsf{EncTree}$ corresponding to $\mathsf{snapshot}_{j_\ell}$.

Basically, the key accumulator still adds new users to the single balanced Merkle tree $\mathsf{EncTree}$ defined before. However, it also stores older snapshots

[3] The snapshotting trick was implicitly used in [48] for similar reasons, which is to build an *efficient* RBE scheme, but their construction instead highlighted the notion of explicitly mapping identities to corresponding timestamps as the more important aspect. We instead choose to focus mostly on the snapshotting principle since it is the major contributor to improving efficiency.

of the EncTree (thereby older snapshots of the registration process). When a new user is added, a tuple $(1, \mathsf{snapshot})$ is added to the list of parameters, where snapshot is the latest description of EncTree. Now older snapshots get replaced with the latest snapshots after new user registration if there exist two different snapshots but for the same group size. By careful analysis and non-trivial execution of the above idea, we were able to show that the resulting RBE scheme is efficient. (By non-trivial execution, we mean that a straightforward implementation/generic usage of balanced Merkle trees lead to an RBE system which is only efficient in the amortized sense, but if the balanced Merkle trees are *lazily* created, then we obtain a fully efficient RBE scheme as desired. More details are provided in the main body.)

7.8.3 Making RBE Verifiable

It turns out that our simplified RBE construction is already very well suited for providing succinct proofs of pre/post-registration. This is due to the fact that the underlying technology being used is a Merkle tree, for which we know how to provide succinct proof of membership. Since the Merkle trees we are building are balanced and sorted, we also can provide succinct proofs of non-membership. Looking ahead, the proofs of pre-registration will consist of proofs of non-membership, and proofs of post-registration would be a combination of proofs of membership and non-membership.

7.8.4 Pre-Registration Proofs

For ease of exposition, consider that the public parameters contain exactly one root node and depth value (rt, d). The idea behind pre-registration proof readily extends to the general case when the public parameters contain more than one root node and depth value pairs. Recall that for soundness of pre-registration verifiability, we need to argue that if the adversary produces an accepting pre-registration proof π for an identity id and parameters pp, then any ciphertext ct encrypted towards id under parameters pp must hide the plaintext. Now we know that in our construction, to decrypt such a ciphertext ct, the adversary must generate a *well-formed* path in the encryption tree $\mathsf{EncTree}$ such that the leaf node contains the identity id.

Here well-formedness of a path (a sequence of nodes from root to a leaf) is formally defined as follows. Let the path under consideration be $\mathsf{path} = (\mathsf{node}_1, \ldots, \mathsf{node}_d)$ where $\mathsf{node}_i = (h_{i,\mathsf{left}}, \mathsf{id}_i, h_{i,\mathsf{right}})$ for all i. We say path is well-formed if the following conditions are satisfied:

1. All the adjacent nodes obey the merkle tree hash constraints, i.e. either $h_{i,\mathsf{left}} = \mathsf{Hash}(\mathsf{hk}, \mathsf{node}_{i+1})$ or $h_{i,\mathsf{right}} = \mathsf{Hash}(\mathsf{hk}, \mathsf{node}_{i+1})$ for all i,

   (this also tells whether node_{i+1} is a left child or a right child of node_i)

2. If node_{i+1} is the left child of node_i, then it must be that $\mathsf{id}_j \leq \mathsf{id}_i$; otherwise $\mathsf{id}_j > \mathsf{id}_i$, (for all $j > i$)

3. Root rt is same as node_1.

Similarly, we define the notion of adjacent paths. For $b \in \{0,1\}$, consider two paths $\mathsf{path}^{(b)} = (\mathsf{node}_1^{(b)}, \ldots, \mathsf{node}_d^{(b)})$ where $\mathsf{node}_i^{(b)} = (h_{i,\mathsf{left}}^{(b)}, \mathsf{id}_i^{(b)}, h_{i,\mathsf{right}}^{(b)})$. For two distinct paths $\mathsf{path}^{(0)}$ and $\mathsf{path}^{(1)}$, we say they are adjacent if the following conditions are satisfied:

1. Paths $\mathsf{path}^{(0)}$ and $\mathsf{path}^{(1)}$ are well-formed,
2. Nodes $\mathsf{node}_{k+1}^{(0)}$ and $\mathsf{node}_{k+1}^{(1)}$ are left and right child (respectively) of nodes $\mathsf{node}_k^{(0)}$ and $\mathsf{node}_k^{(1)}$

   (where k is the largest index such that first k nodes in paths $\mathsf{path}^{(0)}$ and $\mathsf{path}^{(1)}$ are identical)
3. For all $j > k+1$, nodes $\mathsf{node}_j^{(0)}$ and $\mathsf{node}_j^{(1)}$ are right and left child of their respective parent nodes

   (where k is as defined above).

At this point, the pre-registration proofs follow from a natural observation which is that — if some identity id has not yet been registered as per the encryption tree $\mathsf{EncTree}$ (maintained by the key accumulator), then there must exist two identities $\mathsf{id}_{\mathsf{lwr}}$ and $\mathsf{id}_{\mathsf{upr}}$ such that $\mathsf{id}_{\mathsf{lwr}} < \mathsf{id} < \mathsf{id}_{\mathsf{upr}}$ and paths from the root node to leaf nodes containing $\mathsf{id}_{\mathsf{lwr}}$ and $\mathsf{id}_{\mathsf{upr}}$ are adjacent. That is, a pre-registration proof consists of two adjacent paths $\mathsf{path}_{\mathsf{lwr}}$ and $\mathsf{path}_{\mathsf{upr}}$ with identity relations as described above.[4] Such proofs can be very efficiently

[4]In case id is either smaller or larger than all registered identities, then the proof will consist of exactly one path instead of two. Here we ignore that for simplicity.

computed by performing an extended binary search for id, where the extension corresponds to finding the closest registered identities both larger and smaller than id. Note that a verifier can perform the adjacency-check along with the check that the identities are arranged as $\mathsf{id}_{\mathsf{lwr}} < \mathsf{id} < \mathsf{id}_{\mathsf{upr}}$ for verifying the pre-registration proof.

In summary, the idea is that proof of pre-registration for an identity id can be provided using *structured* proofs of membership for two identities $\mathsf{id}_{\mathsf{lwr}}$ and $\mathsf{id}_{\mathsf{upr}}$, where the structure is formalized by the concept of adjacency as described above. The proof of soundness and correctness builds upon the aforementioned intuition. We provide more details later.

7.8.5 Post-Registration Proofs

As in the case for pre-registration proofs, let us focus on the case where the public parameters contain a single root node and depth pair. Recall that an accepting post-registration proof π for identity-key pair $(\mathsf{id}, \mathsf{pk})$ w.r.t parameters pp must guarantee that a key accumulator uniquely added the identity-key pair $(\mathsf{id}, \mathsf{pk})$ to accumulated list of registered users. The post-registration proofs in our construction can also be visualized similarly to the pre-registration proofs.

Specifically, observe that if some identity id has been registered as per the encryption tree $\mathsf{EncTree}$ (maintained by the key accumulator), then there must exist two identities $\mathsf{id}_{\mathsf{lwr}}$ and $\mathsf{id}_{\mathsf{upr}}$ such that $\mathsf{id}_{\mathsf{lwr}} < \mathsf{id} < \mathsf{id}_{\mathsf{upr}}$ and paths

from the root node to leaf nodes containing $\mathsf{id}_{\mathsf{lwr}}$ and id, and id and $\mathsf{id}_{\mathsf{upr}}$ are adjacent. In other words, if identity id was uniquely registered, then there must exist three disjoint paths $\mathsf{path}_{\mathsf{lwr}}, \mathsf{path}_{\mathsf{mid}}$ and $\mathsf{path}_{\mathsf{upr}}$ such that $\mathsf{path}_{\mathsf{lwr}}, \mathsf{path}_{\mathsf{mid}}$ are adjacent as well as $\mathsf{path}_{\mathsf{mid}}, \mathsf{path}_{\mathsf{upr}}$ with the identities in their respective leaf nodes are related as described above.[5] As for pre-registration proofs, the aforementioned post-registration proof can be computed analogously in an efficient manner. The verification procedure can also be naturally extended from pre-registration proof.

There is, however, one important distinction in the case of post-registration proofs. Note that a pre-registration proof w.r.t. public parameters that contain multiple root node and depth pairs consist of independently computed pre-registration proofs for each root node and depth pair. This is because each sub-proof would guarantee that id was not registered as per that corresponding encryption tree snapshot. Thus, together all these sub-proof would guarantee that id was not registered as per any existing encryption tree snapshot. On the other hand, a post-registration proof w.r.t. public parameters with multiple root node and depth pairs will *not* consist of independently computed post-registration proofs for each root node and depth pair. This is because the identity-key pair $(\mathsf{id}, \mathsf{pk})$ may be registered as per only one root node and depth pair (say the latest snapshot). In contrast, it is not registered as per the remaining (older) snapshots. Therefore, a post-registration proof,

[5] As before, incase id is either the smallest or largest registered identity, then the proof will consist of exactly two paths instead of three. Here we ignore that for simplicity.

in this case, will consist of a mixture of post-registration and pre-registration proofs depending upon whether $(\mathsf{id}, \mathsf{pk})$ was registered as per that encryption tree snapshot.

Common Reference String Model. We would like to note that the construction works in the common reference string (CRS) model. That is, we assume that a hash key is sampled honestly by some trusted party and included as part of the common reference string crs. Thus, in our setting, a malicious key accumulator is allowed to arbitrarily perform corruptions, except being allowed to choose the hash-key (i.e., the crs). It is interesting whether Verifiable RBE systems are possible in the standard model. At first glance, it might seem like Verifiable RBE in the standard model might be impossible. It might be possible to extend the impossibility results such as the impossibility of key-less collision-resistant hash functions and SNARGS in the standard model [55] to the VRBE setting as well. However, this needs more research, and we leave it as an interesting open problem.

7.9 A Generic Approach to Verifiability?

A natural question a reader might ask is whether it would be possible to provide proofs of pre/post-registration verifiability generically for any RBE scheme using a succinct non-interactive proof system such as SNARGs/SNARKs [81, 63, 55, 78, 11] for instance. One possible approach along these lines could be to maintain an external sorted hash tree of registered identities. For

providing pre/post-registration proof, the accumulator would generate (non-)membership proofs for the hash tree along with a SNARK for proving the consistency of the external tree w.r.t. the RBE public parameters. Such a generic approach seems possible but would require maintaining additional data structures for consistency checks. More importantly, this approach necessitates making additional assumptions. For most succinct non-interactive proof systems, we either need to make certain non-falsifiable assumptions [82, 55], or work in the Random Oracle model [81, 8]. Our construction and the proofs of verifiability do not rely on any extra assumptions other than what is already required in existing RBE systems [47, 48] themselves. Thus our results show that verifiability comes for free.

Also, note that SNARKs are usually defined for a family of circuits. Thus, the prover's running time is always as large as the size of the circuit. In contrast, our provers already have random-access to the auxiliary information in this case. We provided highly efficient provers whose run time grows only poly-logarithmically with the number of users. Therefore, our non-generic approach is more interesting theoretically and practically since we do not make any non-standard assumptions, nor do we incur additional overhead in the efficiency.

Chapter 8

Background on Registration-based Encryption

In this chapter, we review the definitions of public-key encryption, hash garbling and verifiable registration-based encryption primitives.

8.1 Public Key Encryption

A public key encryption (PKE) scheme PKE for message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. The setup algorithm takes as input the security parameter λ, and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M}_\lambda) \to \mathsf{ct}$. The encryption algorithm takes as input a public key pk and a message m, and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$. The decryption algorithm takes as input a secret key sk and a ciphertext ct, and outputs a message m.

Correctness. A PKE scheme $\mathcal{E}$ for message spaces $\mathcal{M}$ is said to be correct if for all λ, $m \in \mathcal{M}_\lambda$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, we have that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m$.

Security. The standard security notion for PKE schemes is IND-CPA security. Formally, it is defined as follows.

Definition 8.1.1. *A public key encryption scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA secure if for every stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$*, such that the following holds*

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\end{array}\right] \leq \frac{1}{2} + negl(\lambda).$$

8.2 Hash Garbling

We now review the notion of hash garbling scheme introduced in [47]. The construction of hash garbling from a one-way function with encryption has been implicit in many works [30, 38, 42, 40, 20, 47]. Garg et al. [48] also showed an explicit construction of hash garbling from batch encryption (which itself can be built from a one-way function with encryption).

$\mathsf{Setup}(1^\lambda, 1^\ell) \to \mathsf{hk}$. The setup algorithm takes as input the security parameter λ, an input length parameter ℓ, and outputs a hash key hk.

$\mathsf{Hash}(\mathsf{hk}, x) \to y$. This is a deterministic algorithm that takes as input a hash key hk and a value $x \in \{0,1\}^\ell$ and outputs a value $y \in \{0,1\}^\lambda$.

$\mathsf{GarbleCkt}(\mathsf{hk}, C, \mathsf{state}) \to \tilde{C}$**.** It takes as input hash key hk, a circuit C, a secret state $\mathsf{state} \in \{0,1\}^\lambda$ and outputs a garbled circuit $\tilde{C}$.

$\mathsf{GarbleInp}(\mathsf{hk}, y, \mathsf{state}) \to \tilde{y}$. It takes as input hash key hk, a value $y \in \{0,1\}^\lambda$, a secret state $\mathsf{state} \in \{0,1\}^\lambda$ and outputs a garbled value $\tilde{y}$.

$\mathsf{Eval}(\tilde{C}, \tilde{y}, x) \to z$. This takes as input a garbled circuit $\tilde{C}$, a garbled value $\tilde{y}$, a value $x \in \{0,1\}^\ell$ and outputs a value z.

We now state the correctness and security requirements for a hash garbling scheme.

Correctness. A hash garbling scheme is said to be correct if for all $\lambda \in \mathbb{N}$, $\ell \in \mathbb{N}$, hash key $\mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, circuit C, input $x \in \{0,1\}^\ell$, $\mathsf{state} \in \{0,1\}^\lambda$, garbled circuit $\tilde{C} \leftarrow \mathsf{GarbleCkt}(\mathsf{hk}, C, \mathsf{state})$ and a garbled value $\tilde{y} \leftarrow \mathsf{GarbleInp}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, x), \mathsf{state})$, we have $\mathsf{Eval}(\tilde{C}, \tilde{y}, x) = C(x)$.

Security. We now define the security requirement for hash garbling scheme.

Definition 8.2.1. *A hash garbling scheme is said to be secure if there exists a PPT simulator* Sim *such that for every stateful PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for every* $\lambda, \ell \in \mathbb{N}$*, we have*

$$\Pr\left[\mathcal{A}(\tilde{C}_b, \tilde{y}_b) = b \;:\; \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); (C, x) \leftarrow \mathcal{A}(\mathsf{hk}); \mathsf{state} \leftarrow \{0,1\}^\lambda \\ \tilde{C}_0 \leftarrow \mathsf{GarbleCkt}(\mathsf{hk}, C, \mathsf{state}); \tilde{y}_0 \leftarrow \mathsf{GarbleInp}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, x), \mathsf{state}) \\ (\tilde{C}_1, \tilde{y}_1) \leftarrow \mathsf{Sim}(\mathsf{hk}, x, 1^{|C|}, C(x)); b \leftarrow \{0,1\} \end{array}\right] \leq \frac{1}{2} +$$

8.3 Verifiable Registration-based Encryption

In this section, we define the notion of Verifiable Registration Based Encryption (VRBE). First, we recall the definition of Registration Based Encryption (RBE) as introduced in [47]. For message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ and identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_\lambda$, an RBE system consists of the following algorithms —

$\mathsf{CRSGen}(1^\lambda) \to \mathsf{crs}$. The CRS generation algorithm takes as input the security parameter λ, and outputs a common reference string crs.

$\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. The key generation algorithm takes as input the security parameter 1^λ, and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$. (Note that these are only public and secret keys, not the encryption/decryption keys.)

$\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$. The registration algorithm is a deterministic algorithm, that takes as input the common reference string crs, current public parameter pp, an identity id to be registered, and a corresponding public key pk. It maintains auxiliary information aux, and outputs the updated parameters pp'. The registration algorithm is modelled as a RAM program where it can read/write to arbitrary locations of the auxiliary information aux. (The system is initialized with pp and aux set to ϵ.)

$\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the common reference string crs, public parameters pp, a recipient identity id, and a plaintext message m. It outputs a ciphertext ct.

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to u$. The key update algorithm is a deterministic algorithm, that takes as input the public parameters pp and an identity id. Given the auxiliary information aux, it generates the key update $u \in \{0,1\}^*$. Similar to the registration algorithm, this is also modelled as a RAM

program, but it is only given read access to arbitrary locations of the auxiliary information aux.

$\mathsf{Dec}(\mathsf{sk}, u, \mathsf{ct}) \to m/\mathsf{GetUpd}/\perp$. The decryption algorithm takes as input a secret key sk, a key update u, and a ciphertext ct, and it outputs either a message $m \in \mathcal{M}$, or a special symbol in $\{\perp, \mathsf{GetUpd}\}$. (Here GetUpd indicates that a key update might be needed for decryption.)

Next, we introduce the notion of verifiability for an RBE system. Here we consider the notions of pre-registration as well as post-registration verifiability. Intuitively, the goal of pre-registration verifiability is to provide a short proof validating that a given id has not yet been registered. Any ciphertext encrypted towards such an identity will completely hide the message even if all other secret keys are leaked. Similarly, the intuition behind post-registration verifiability is to provide a short proof of unique addition, where the proof guarantees that the key accumulator (i.e., the party responsible for registration) must not have added a trapdoor during a possibly dishonest registration which allows decryption of ciphertexts intended for that particular user. Formally, we introduce four new algorithms — $\mathsf{PreProve}, \mathsf{PreVerify}, \mathsf{PostProve}, \mathsf{PostVerify}$ with the following syntax:

$\mathsf{PreProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}) \to \pi$. The pre-registration prover algorithm is deterministic, which takes as input the common reference string crs, public parameters pp, and identity id. Given the auxiliary information aux, it outputs a pre-registration proof π. Like the registration algorithm, this

is also modeled as a RAM program. However, it is only given read access to arbitrary locations of the auxiliary information aux.

$\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) \to 0/1$. The pre-registration verifier algorithm takes as input the common reference string crs, public parameters pp, an identity id, and a proof π. It outputs a single bit $0/1$, denoting whether the proof is accepted or not.

$\mathsf{PostProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \pi$. The post-registration prover algorithm is deterministic. It takes as input the common reference string crs, public parameters pp, an identity id, and a public key pk. Given the auxiliary information aux, it outputs a post-registration proof π. Like the registration algorithm, this is also modeled as a RAM program. However, it is only given read access to arbitrary locations of the auxiliary information aux.

$\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) \to 0/1$. The post-registration verifier algorithm takes as input the common reference string crs, public parameters pp, an identity id, a public key pk, and a proof π. It outputs a single bit $0/1$, denoting whether the proof is accepted or not.

If one does not impose any succinctness requirements on the pre/post-registration proofs, then the above algorithms are directly implied because the registration process is deterministic. This is because the proofs themselves can set to be the auxiliary information aux, and one could perform verification

by simply rebuilding the public parameters given in aux. This is quite inefficient. Thus we impose succinctness restrictions along with completeness and soundness restrictions on the pre/post-registration.

Remark 8.3.1. *We consider deterministic provers both for pre-registration and post-registration proofs in the above abstraction. Although one could instead consider randomized proving algorithms, we avoid it since our construction already achieves deterministic proving. This also leads to simpler correctness and security definitions. Looking ahead, in all our security and correctness definitions, we do not provide the adversary any oracle queries to the* $\mathsf{PreProve}$ *and* $\mathsf{PostProve}$ *algorithms since they are deterministic given access to the auxiliary information* aux*. Since the adversary can itself maintain auxiliary information and proof algorithms are deterministic, oracle queries are redundant.*

8.3.1 Correctness

Below we first recall the definition of completeness, compactness, and efficiency for RBE systems as studied previously. After that, we introduce the definitions for completeness, compactness, and efficiency of the pre/post-registration processes.

Definition 8.3.1 (Completeness, compactness, and efficiency of RBE)**.** *For any (stateful) interactive computationally unbounded adversary* $\mathcal{A}$ *that has a* $\mathsf{poly}(\lambda)$ *round complexity, consider the following game* $\mathsf{Comp}^{\mathsf{RBE}}_{\mathcal{A}}(\lambda)$*.*

1. *(Initialization) The challenger initializes parameters as* $(\mathsf{pp}, \mathsf{aux}, u, S_{\mathsf{ID}}, \mathsf{id}^*, t) = (\epsilon, \epsilon, \epsilon, \emptyset, \perp, 0)$*, samples* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$*, and sends the* crs *to* $\mathcal{A}$*.*

2. *(Query Phase)* $\mathcal{A}$ *makes polynomially many queries of the following form, where each query is considered as a single round of interaction between the challenger and the adversary.*

   (a) ***Registering new (non-target) identity.*** *On a query of the form* $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$*, the challenger checks that* $\mathsf{id} \notin S_{\mathsf{ID}}$*, and registers* $(\mathsf{id}, \mathsf{pk})$ *by running the registration procedure as* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$*. It adds* id *to the set as* $S_{\mathsf{ID}} := S_{\mathsf{ID}} \cup \{\mathsf{id}\}$*. (Note that the challenger updates the parameters* $\mathsf{pp}, \mathsf{aux}, S_{\mathsf{ID}}$ *after each query. Also, it aborts if the check fails.)*

   (b) ***Registering target identity.*** *On a query of the form* $(\mathsf{regtgt}, \mathsf{id})$*, the challenger first checks that* $\mathsf{id}^* = \perp$*. If the check fails, it aborts. Else, it sets* $\mathsf{id}^* := \mathsf{id}$*, samples challenge key pair* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$*, updates public parameters as* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$*, and sets* $S_{\mathsf{ID}} := S_{\mathsf{ID}} \cup \{\mathsf{id}^*\}$*. Finally, it sends the challenge public key* pk^* *to* $\mathcal{A}$*. (Here the challenger stores the secret key* sk^* *in addition to updating all other parameters. Also, note that the adversary here is restricted to make such a query at most once, since the challenger would abort otherwise.)*

   (c) ***Target identity encryptions.*** *On a query of the form* (enctgt, m)*, the challenger checks if* $\mathsf{id}^* \neq \perp$*. It aborts if the check fails. Oth-*

erwise, it sets $t := t+1$, $\widetilde{m}_t := m$, *and computes ciphertext* $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \widetilde{m}_t)$. *It stores the tuple* $(t, \widetilde{m}_t, \mathsf{ct}_t)$, *and sends the ciphertext* ct_t *to* $\mathcal{A}$.[1]

(d) ***Target identity decryptions.*** *On a query of the form* (dectgt, j), *the challenger checks if* $\mathsf{id}^* \neq \perp$ *and* $j \in [t]$. *It aborts if the check fails. Otherwise, it computes* $y_j = \mathsf{Dec}(\mathsf{sk}^*, u, \mathsf{ct}_j)$. *If* $y_j = \mathsf{GetUpd}$, *then it generates the fresh update* $u := \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ *and then recomputes* $y_j = \mathsf{Dec}(\mathsf{sk}^*, u, \mathsf{ct}_j)$. *Finally, the challenger stores the tuple* (j, y_j).

3. *(Output Phase) We say that the adversary* $\mathcal{A}$ *wins the game if there is some* $j \in [t]$ *for which* $\widetilde{m}_j \neq y_j$.

Let $n = |S_{\mathsf{ID}}|$ *denote the number of identities registered until any specific round in the above game. We say that an RBE scheme is complete, compact, and efficient if for every (stateful) interactive computationally unbounded adversary* $\mathcal{A}$ *that has a* $\mathsf{poly}(\lambda)$ *round complexity, there exists polynomials* p_1, p_2, p_3, p_4, p_5 *and a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$, *the following holds:*

Completeness. $\Pr[\mathcal{A}$ *wins in* $\mathsf{Comp}^{\mathsf{RBE}}_{\mathcal{A}}(\lambda)] \leq negl(\lambda)$.

[1]Here and throughout, whenever we write the challenger stores the tuple, we mean that it appends this to its local state such that these could be obtained by the challenger when referred to later in the game.

Compactness of public parameters and updates. $|\mathsf{pp}| \leq p_1(\lambda, \log n)$ *and* $|u| \leq p_2(\lambda, \log n)$.

Efficiency of registration and update. *The running time of each invocation of* Reg *and* Upd *algorithms is at most* $p_3(\lambda, \log n)$ *and* $p_4(\lambda, \log n)$*, respectively. (Note that this implies the above compactness property.)*

Efficiency of the number of updates. *The total number of invocations of* Upd *for identity* id^* *during* target identity decryption phase *(i.e., Step* 2d *of game* $\mathsf{Comp}_{\mathcal{A}}^{\mathsf{RBE}}(\lambda)$*) is at most* $p_5(\lambda, \log n)$ *for every* n.

Next, we introduce the completeness, compactness, and efficiency conditions we require from the pre/post-registration procedures of a Verifiable RBE system. Briefly, the completeness of $(\mathsf{PreProve}, \mathsf{PreVerify})$ algorithms states that for any identity id^* that has not yet been registered, the key accumulator should be able to compute a proof π (by running the $\mathsf{PreProve}$ algorithm) such that proof π guarantees id^* has not yet been registered. Similarly, for the post-registration verification, the completeness of $(\mathsf{PostProve}, \mathsf{PostVerify})$ algorithms states that for any identity id^* that has been (honestly) registered, the key accumulator should be able to compute a proof π (by running the $\mathsf{PostProve}$ algorithm) such that proof π guarantees id^* has been registered.

In addition to the above natural completeness definition for the post-registration verification, we also define a stronger completeness property that provides a certain extractability guarantee. Informally, it states that if there exists a post-registration proof π for identity-key pair $(\mathsf{id}^*, \mathsf{pk}^*)$ that is accepted

by the $\mathsf{PostVerify}$ algorithm, then every honestly generated ciphertext intended towards id^* can be decrypted by the corresponding secret key sk^* and some update u. Here the update u is (publicly, efficiently, and deterministically) computable from the proof π itself, instead of the auxiliary information aux.

Definition 8.3.2 (Completeness, compactness, and efficiency of Pre/Post-Registration Proofs)**.** *For any (stateful) interactive computationally unbounded adversary $\mathcal{A}$ that has a $\mathsf{poly}(\lambda)$ round complexity, consider the following game $\mathsf{Comp}_{\mathcal{A}}^{\mathsf{VRBE}}(\lambda)$.*

1. *(Initialization) The challenger initializes parameters as $(\mathsf{pp}, \mathsf{aux}, S_{\mathsf{ID}}, S_{\mathsf{ID},\mathsf{PK}}, t) = (\epsilon, \epsilon, \emptyset, \emptyset, 0)$, samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$, and sends the crs to $\mathcal{A}$.*

2. *(Query Phase) $\mathcal{A}$ makes polynomially many queries of the following form, where each query is considered as a single round of interaction between the challenger and the adversary.*

   (a) ***Registering new identity.*** *On a query of the form $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$, the challenger checks that $\mathsf{id} \notin S_{\mathsf{ID}}$, and registers $(\mathsf{id}, \mathsf{pk})$ by running the registration procedure as $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$. It adds id and pk to the sets $S_{\mathsf{ID}}, S_{\mathsf{ID},\mathsf{PK}}$ as $S_{\mathsf{ID}} := S_{\mathsf{ID}} \cup \{\mathsf{id}\}$ and $S_{\mathsf{ID},\mathsf{PK}} := S_{\mathsf{ID},\mathsf{PK}} \cup \{(\mathsf{id}, \mathsf{pk})\}$.*

   (b) ***Pre-registration proofs.*** *On a query of the form $(\mathsf{prereg}, \mathsf{id})$, the challenger checks if $\mathsf{id} \notin S_{\mathsf{ID}}$. It aborts if the check fails. Otherwise, it sets $t := t+1$, and computes the proof $\pi_t^{\mathsf{pre}} = \mathsf{PreProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id})$. Next, it verifies the proof π_t^{pre} as $\beta_t = \mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi_t^{\mathsf{pre}})$, and stores the tuple $(\mathsf{prereg}, t, \beta_t)$.*

(c) ***Post-registration proofs.*** *On a query of the form* $(\mathsf{postreg}, \mathsf{id}, \mathsf{pk})$*, the challenger checks if* $(\mathsf{id}, \mathsf{pk}) \in S_{\mathsf{ID},\mathsf{PK}}$*. It aborts if the check fails. Otherwise, it sets* $t := t + 1$*, and computes the proof* $\pi_t^{\mathsf{post}} = \mathsf{PostProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$*. Next, it verifies the proof* π_t^{post} *as* $\beta_t = \mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi_t^{\mathsf{post}})$*, and stores the tuple* $(\mathsf{postreg}, t, \beta_t)$*.*

3. *(Output Phase) We say that the adversary* $\mathcal{A}$ *wins the game if for some* $j \in [t]$ *there exists a tuple of the form* $(\mathsf{prereg}, j, 0)$ *or* $(\mathsf{postreg}, j, 0)$*. (That is, there exists proof that does not verify.)*

Let $n = |S_{\mathsf{ID}}|$ *denote the number of identities registered until any specific round in the above game. We say that a VRBE scheme achieves pre/post-registration completeness, compactness, and efficiency if for every (stateful) interactive computationally unbounded adversary* $\mathcal{A}$ *that has a* $\mathsf{poly}(\lambda)$ *round complexity, there exists polynomials* p_1, p_2, p_3, p_4, p_5 *and a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

Completeness. $\Pr[\mathcal{A}$ *wins in* $\mathsf{Comp}_{\mathcal{A}}^{\mathsf{VRBE}}(\lambda)] \leq negl(\lambda)$*.*

Compactness of proofs. $|\pi_t^{\mathsf{pre}}| \leq p_1(\lambda, \log n)$ *and* $|\pi_t^{\mathsf{post}}| \leq p_2(\lambda, \log n)$*.*

Efficiency of provers. *The running time of each invocation of* $\mathsf{PreProve}$ *and* $\mathsf{PostProve}$ *algorithms is at most* $p_3(\lambda, \log n)$ *and* $p_4(\lambda, \log n)$*, respectively. (Note that this implies the above compactness property.)*

Next, we define the stronger extractable completeness for post-registration proofs.

Definition 8.3.3 (Efficiently Extractable Completeness for Post-Registration)**.** *A VRBE scheme satisfies efficiently extractable completeness property for post-registration if there exists a deterministic polynomial-time algorithm* UpdExt *such that for any computationally unbounded admissible adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, u, \mathsf{ct}) \neq m \ : \ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{pp}, \mathsf{id}, \pi, m) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{pk}), u \leftarrow \mathsf{UpdExt}(\pi, \mathsf{id}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \end{array}\right] \leq negl(\lambda)$$

where an adversary $\mathcal{A}$ *is said to be admissible if it always produces a valid post-registration proof* π *i.e.,* $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) = 1$.

8.3.2 Security

We first recall the security definition for RBE systems as studied previously. After that, we introduce the soundness definitions for the pre/post-registration proofs.

Definition 8.3.4 (Message Hiding Security)**.** *For any (stateful) interactive PPT adversary* $\mathcal{A}$*, consider the following game* $\mathsf{Sec}_{\mathcal{A}}^{\mathsf{RBE}}(\lambda)$.

1. *(Initialization) The challenger initializes parameters as* $(\mathsf{pp}, \mathsf{aux}, u, S_{\mathsf{ID}}, \mathsf{id}^*) = (\epsilon, \epsilon, \epsilon, \emptyset, \bot)$*, samples* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$*, and sends the* crs *to* $\mathcal{A}$.

2. *(Query Phase)* $\mathcal{A}$ *makes polynomially many queries of the following form:*

(a) ***Registering new (non-target) identity.*** *On a query of the form* $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$*, the challenger checks that* $\mathsf{id} \notin S_{\mathsf{ID}}$*, and registers* $(\mathsf{id}, \mathsf{pk})$ *by running the registration procedure as* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$*. It adds* id *to the set as* $S_{\mathsf{ID}} := S_{\mathsf{ID}} \cup \{\mathsf{id}\}$*.*

(b) ***Registering target identity.*** *On a query of the form* $(\mathsf{regtgt}, \mathsf{id})$*, the challenger first checks that* $\mathsf{id}^* = \perp$*. If the check fails, it aborts. Else, it sets* $\mathsf{id}^* := \mathsf{id}$*, samples challenge key pair* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$*, updates public parameters as* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$*, and sets* $S_{\mathsf{ID}} := S_{\mathsf{ID}} \cup \{\mathsf{id}^*\}$*. Finally, it sends the challenge public key* pk^* *to* $\mathcal{A}$*.*

3. *(Challenge Phase) On a query of the form* $(\mathsf{chal}, \mathsf{id}, m_0, m_1)$*, then the challenger checks if* $\mathsf{id} \notin S_{\mathsf{ID}} \backslash \{\mathsf{id}^*\}$*. It aborts if the check fails. Otherwise, it samples a bit* $b \leftarrow \{0, 1\}$ *and computes challenge ciphertext* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$*.*

4. *(Output Phase) The adversary* $\mathcal{A}$ *outputs a bit* b' *and wins the game if* $b' = b$*.*

We say that an RBE scheme is message-hiding secure if for every (stateful) interactive PPT adversary $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*,* $\Pr[\mathcal{A}$ *wins in* $\mathsf{Sec}^{\mathsf{RBE}}_{\mathcal{A}}(\lambda)] \leq \frac{1}{2} + negl(\lambda)$*.*

Finally, we define the soundness requirements for our pre/post-registration proof systems. Informally, the pre-registration soundness states that any adversarial key accumulator must not be able to simultaneously — 1) provide

a valid (acceptable) proof of pre-registration for some identity id, 2) able to break semantic security for (honestly generated) ciphertexts intended towards identity id. Intuitively, this says that even a corrupt key accumulator must not decrypt ciphertexts intended for unregistered users while providing an accepting pre-registration proof. Thus, any new user can ask for pre-registration proof to verify that the key accumulator has not inserted any trapdoor that enables the accumulator to decrypt ciphertexts encrypted for that user.

In a similar vein, the post-registration soundness informally states that any adversarial key accumulator must not be able to simultaneously — 1) provide a valid (acceptable) proof of post-registration for some identity-key pair $(\mathsf{id}, \mathsf{pk})$ (where pk has honestly generated and the associated secret key was not revealed), 2) able to break semantic security for (honestly generated) ciphertexts intended towards identity id. Intuitively, this says that even a corrupt key accumulator must not be able to decrypt ciphertexts intended for registered users while being able to provide an accepting post-registration proof. Thus, any registered user can ask for post-registration proof to verify that the key accumulator has not inserted any trapdoor that enables the accumulator to decrypt ciphertexts encrypted for that user. Now we give the formal definitions.

Definition 8.3.5 (Soundness of Pre-Registration Verifiability)**.** *A VRBE scheme satisfies the soundness of pre-registration verifiability if, for every stateful* admissible *PPT adversary $\mathcal{A}$, there is a negligible function $negl(\cdot)$ such that for*

every $\lambda \in \mathbb{N}$, the following holds.

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b \ : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda) \\ (\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs}) \\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b) \end{array}\right] \leq \frac{1}{2} + \mathit{negl}(\lambda),$$

where $\mathcal{A}$ is admissible if and only if π is a valid pre-registration proof, i.e. $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) = 1$.

Definition 8.3.6 (Soundness of Post-Registration Verifiability)**.** *A VRBE scheme satisfies the soundness of post-registration verifiability if, for every stateful* admissible *PPT adversary $\mathcal{A}$, there is a negligible function $\mathit{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds.*

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b \ : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda);\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{pk}) \\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b) \end{array}\right] \leq \frac{1}{2} + \mathit{negl}(\lambda),$$

where $\mathcal{A}$ is admissible if and only if π is a valid post-registration proof, i.e. $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) = 1$.

Chapter 9

Verifiable RBE from Standard Assumptions

In this chapter, we present our verifiable RBE construction and prove its security. Our construction relies on two primitives — a regular PKE scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, and a hash garbling scheme $\mathsf{HG} = (\mathsf{HG.Setup}, \mathsf{HG.Hash}, \mathsf{HG.GarbleCkt}, \mathsf{HG.GarbleInp}, \mathsf{HG.Eval})$. Below we provide a detailed outline of our construction.

9.1 Construction

For ease of exposition, we assume that the length of identities supported, length of public keys generated by Gen algorithm, the output length of the hash is λ-bits, and the input length of the hash function is $(3\lambda+1)$-bits. Note that this can be avoided by simply selecting parameters accordingly. Below we define some useful notation that we will reuse throughout the sequel. Additionally, we describe how to interpret the auxiliary information and the public parameters in our construction.

Abstractions, Trees, and Notations. In our construction, the key accumulator maintains two types of balanced binary trees. The first tree, which we

refer to as the IDTree is a balanced binary tree in which each node has a label of the form $(\mathsf{id}, t) \in \{0,1\}^{2\lambda}$, and the nodes are basically being sorted as per the first tuple entry which is id. (Concretely, $(\mathsf{id}_1, t_1) \prec (\mathsf{id}_2, t_2)$ iff $\mathsf{id}_1 < \mathsf{id}_2$, where $\prec$ denotes the node ordering.) This tree is used as an internal storage object (which provides fast node insertion/lookup) by the key accumulator. Here id denotes the registered identity, and t denotes the timestamp (i.e., number of users already registered $+1$).

The second family of trees which we refer to are the *encryption* trees $\{\mathsf{EncTree}_i\}_{i \in [\ell_n]}$ for some $\ell_n > 0$. Each such tree consists of two-types of nodes — (1) leaf nodes which store a registered identity-key pair, (2) non-leaf nodes which store the hash values of its children and largest registered identity in its left sub-tree. Concretely, each node in the tree has a label of the form $(\mathsf{flag}||a||\mathsf{id}||b) \in \{0,1\}^{3\lambda+1}$. For a leaf node $\mathsf{flag} = 1$, $a = 0^\lambda$, $b = \mathsf{pk}$ and $(\mathsf{id}, \mathsf{pk})$ is identity-key pair of the corresponding registered user. For a non-leaf node, $\mathsf{flag} = 0$, and id denotes the largest registered identity in its left sub-tree, a and b are the hash value of its left and right child's label (respectively). The leaf nodes are inserted as per their registered identity (i.e., the nodes are ordered with an increasing ordering amongst the identities). Concretely, a new leaf node $(1||0^\lambda||\mathsf{id}||\mathsf{pk})$ is added as follows —

1. Perform a binary search, by using the 'largest registered identity in the left sub-tree' information stored in the label of each intermediate node, to find the leaf node with the smallest identity $\widetilde{\mathsf{id}}$ such that $\widetilde{\mathsf{id}} > \mathsf{id}$. (Let $\widetilde{\mathsf{pk}}$ be the key associated with $\widetilde{\mathsf{id}}$.)

2. Delete the leaf node associated with $\widetilde{\mathsf{id}}$, and replace it with a new intermediate node such that $(1||0^{\lambda}||\mathsf{id}||\mathsf{pk})$ and $(1||0^{\lambda}||\widetilde{\mathsf{id}}||\widetilde{\mathsf{pk}})$ are its left and right children (respectively).

3. Perform the *re-balance* operation on the binary tree.[1]

4. Re-compute the labels for all intermediate nodes which have been re-balanced (i.e., moved around). This involves updating the largest registered identity in the left sub-tree information as well as re-computing the corresponding hash values.

Looking ahead, here the first $\ell_n - 1$ encryption trees $\mathsf{EncTree}_1, \ldots, \mathsf{EncTree}_{\ell_n - 1}$ represent the older snapshots of the registration process, whereas $\mathsf{EncTree}_{\ell_n}$ represents the latest encryption tree which contains all the identities registered so far. The above tree insertion operation is efficient ($O(\log n)$ updates and running time) as long as the underlying tree abstraction provides efficient lookup and insertion. Since we use a balanced tree as the underlying abstraction, efficiency follows.

A very useful piece of notation in our scheme is the notion of *'paths'* from the root node to a leaf node in some encryption tree $\mathsf{EncTree}$. Concretely, throughout this chapter, we will define a *path* w.r.t. a tree $\mathsf{EncTree}$

[1]Note that the tree re-balancing operation has to be carefully performed as in our abstraction (as well as the [48] abstraction) the leaf nodes and intermediate nodes are not exchangeable. Thus, the leaf-nodes must always stay the leaf nodes. Roughly one might consider that the re-balancing operation is only performed on the tree obtained by removing all leaf-nodes. This is not completely accurate but captures the underlying intuition.

(with root rt and depth d) as a sequence of (at most) d nodes where the first node is the root node of the tree and last node is a leaf node with certain specific properties. Concretely, any path path will look like $\mathsf{path} = (\mathsf{node}_1, \ldots, \mathsf{node}_{d-1}, \mathsf{node}_d)$, where for $i < d$, $\mathsf{node}_i = (0||a_i||\mathsf{id}_i||b_i)$ for some hash values a_i, b_i and identity id_i. Similarly, $\mathsf{node}_d = (1||0^\lambda||\mathsf{id}_d||\mathsf{pk})$ for some identity-key pair $\mathsf{id}_d, \mathsf{pk}$, and the remaining intermediate nodes are such that for every i, $a_i = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i+1})$ if node_{i+1} is left child of node_i, else $b_i = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i+1})$. Also, if node_{i+1} is left child of node_i then $\mathsf{id}_i \geq \mathsf{id}_{i+1}$, else $\mathsf{id}_i < \mathsf{id}_{i+1}$. Now note that such a path can be efficiently computed for every identity id, which has been added to encryption tree $\mathsf{EncTree}$, by simply performing an extended binary search. We will be re-using this fact many times throughout the sequel.

Lastly, we define a notion which we refer to as *'adjacent'* paths. This is extremely useful for verifiability of our scheme. Note that if during binary search in any balanced search tree, if the node/label that is being searched does not exist, then one could prove that efficiently by giving two paths to nodes with labels that are just bigger than and smaller than the label as per the ordering defined in the tree. More formally, for any two paths path_1 and path_2 in an encryption tree, we can perform an adjacency check efficiently as follows. Let $\mathsf{path}_j = (\mathsf{node}_{1,j}, \ldots, \mathsf{node}_{d-1,j}, \mathsf{node}_{d,j})$ for $j \in [2]$ where $\mathsf{node}_{i,j} = (0||a_{i,j}||\mathsf{id}_{i,j}||b_{i,j})$ for $j < d$ and $\mathsf{node}_{d,j} = (1||0^\lambda||\mathsf{id}_{d,j}||\mathsf{pk}_{d,j})$: (1) First, check that both paths are valid. Note that path validity is checked as

that either $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i+1,j})$ is equal to $a_{i,j}$ or $b_{i,j}$.[2] (2) Next, the verifier first computes the largest common prefix of nodes in paths path_1 and path_2. That is, let k be the largest index such that $\mathsf{node}_{i,1} = \mathsf{node}_{i,2}$ for all $i \leq k$. Now if $\mathsf{id}_{d,1} < \mathsf{id}_{d,2}$, then check that $\mathsf{node}_{k+1,1}$ and $\mathsf{node}_{k+1,2}$ are left and right children of $\mathsf{node}_{k,1} = \mathsf{node}_{k,2}$. Next, it must check that, for all $i > k+1$, $\mathsf{node}_{i,1}$ is always the right child of its parent and $\mathsf{node}_{i,2}$ is always the left child of its parent. Basically, this is done to make sure that these two paths are adjacent and there does not exist any intermediate registered identity between these.

Construction. The key accumulator initializes the public parameters pp and auxiliary information aux as empty strings ϵ. Furthermore, afterward, at any point, the auxiliary information will contain the IDTree and (at most a λ number of) encryption trees $\mathsf{EncTree}_i$ along with a number n_i.[3] And, the public parameters pp consists of root value and depth pairs (rt_i, d_i) for each encryption tree $\mathsf{EncTree}_i$ present in auxiliary information aux. Here rt_i is the root node and d_i is the depth of $\mathsf{EncTree}_i$. We now formally describe our construction.

$\mathsf{CRSGen}(1^\lambda) \to \mathsf{crs}$. The CRS generation algorithm samples a hash key for the hash garbling scheme as $\mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$, and outputs $\mathsf{crs} = \mathsf{hk}$.

[2]Note that this also tells whether $\mathsf{node}_{i+1,j}$ is a left child of $\mathsf{node}_{i,j}$, or right child.

[3]Looking ahead, the number n_i signifies the number of users who will refer to the tree $\mathsf{EncTree}_i$ for decryption. The significance of n_i will become clear in the construction.

$\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and $\mathsf{aux} = \left(\mathsf{IDTree}, \{(\mathsf{EncTree}_i, n_i)\}_{i\in[\ell_n]}\right)$. Also, let $n = \sum_i n_i + 1$. The key accumulator performs the following operations:

1. It creates a leaf node with the label $(1||0^\lambda||\mathsf{id}\,||\mathsf{pk})$, and update the current (latest) encryption tree $\mathsf{EncTree}_{\ell_n}$ by inserting the new leaf node. (Note that the insertion is performed as described above, and it involves balancing the tree and updating the hash values accordingly.)

2. Let $\mathsf{NewTree}$ be the new encryption tree. It continues by adding (id, n) to the IDTree, and the tuple $(\mathsf{EncTree}_{\ell_n+1}, 1) := (\mathsf{NewTree}, 1)$ to current auxiliary information aux. (This new tuple should be interpreted as signifying that only one user (which is the current, i.e. n^{th}, user with identity id) would refer to the latest encryption tree $\mathsf{NewTree}$ during decryption.)

3. Next it modifies the list of encryption trees as follows. Let $\mathsf{aux} = \left(\mathsf{IDTree}, \{(\mathsf{EncTree}_i, n_i)\}_{i\in[\ell_n+1]}\right)$, and

$$\delta = \max\left(\{0\} \cup \left\{i \in [\ell_n - 1] \;:\; \forall\, j \in [i],\; n_{\ell_n+1-j} = 2^{j-1}\right\}\right).$$

It modifes the auxilliary information as $\mathsf{aux} = \left(\mathsf{IDTree}, \{(\mathsf{EncTree}'_i, n'_i)\}_{i\in[\ell_n+1-\delta]}\right)$, where

$$(\mathsf{EncTree}'_i, n'_i) := \begin{cases} (\mathsf{EncTree}_i, n_i) & \text{if } i < \ell_n + 1 - \delta, \\ (\mathsf{NewTree}, 2 \cdot n_i) & \text{otherwise.} \end{cases}$$

In other words, the accumulator removes all the old versions of the encryption trees as long as it could replace all of them with the latest tree until the number of users, which would then refer to the latest tree, stays at a power of 2. To illustrate this operation, we give a detailed running example of the Reg algorithm in Figure 9.1.

4. Lastly, the accumulator modifes the public parameters to $\mathsf{pp}' = \{(\mathsf{rt}'_i, d'_i)\}_{i\in[\ell_n+1-\delta]}$, where rt'_i, d'_i are root node and depth of the encryption tree $\mathsf{EncTree}'_i$ (respectively).

   Note. At a high level, the accumulator maintains the invariant that the i^{th} encryption tree $\mathsf{EncTree}'_i$ is an accumulation of the identity-key pairs for exactly the first $\sum_{j\le i} n'_j$. This tree is intended to be precisely used during decryption by those n'_i users who registered just after the first $\sum_{j\le i-1} n'_i$ users. Additionally, the n_i values for the last and the second last encryption trees are more than a factor of 2 apart. (The last point is crucial in ensuring that the number of updates grows only logarithmically.)

$\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and $\mathsf{crs} = \mathsf{hk}$. The encryptor proceeds as follows:

1. First, it samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda$ and $r_{i,j} \leftarrow \{0,1\}^\lambda$ for each $i \in [\ell_n]$, and $j \in [d_i + 1]$.

2. Next, for each encryption tree $\mathsf{EncTree}_i$, it computes a sequence of d_i hash-garbled circuits as follows:

Sample Execution of Reg Algorithm

Consider the scenario where 7 users $(\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_3, \mathsf{id}_4, \mathsf{id}_5, \mathsf{id}_6, \mathsf{id}_7)$ are registered into the system. The auxiliary information aux now stores IDTree and 3 versions of $\mathsf{EncTree}$. IDTree consists of all the identites along with their timestamps. $\mathsf{EncTree}_1, \mathsf{EncTree}_2$ are the versions of $\mathsf{EncTree}$ when only 4 users and 6 users were registered respectively. $\mathsf{EncTree}_3$ is the latest version of the $\mathsf{EncTree}$ when all 7 users are registered in the system. More precisely, the list of identities present in each $\mathsf{EncTree}_i$ is as follows.

$$\mathsf{aux} = \{\mathsf{IDTree}, (\mathsf{EncTree}_1, 4) : [\mathsf{id}_1, \ldots, \mathsf{id}_4],\ (\mathsf{EncTree}_2, 2) : [\mathsf{id}_1, \ldots, \mathsf{id}_6],\ (\mathsf{EncTree}_3, 1) : [\mathsf{id}_1, \ldots, \mathsf{id}_7]\}$$

Let us now look at when we register a new identity id_8. The key accumulator sets $n = 8$, inserts id_8 into IDTree, creates $\mathsf{NewTree}$ by inserting id_8 into $\mathsf{EncTree}_3$, and sets $(\mathsf{EncTree}_4, 1) = (\mathsf{NewTree}, 1)$. To compute δ, the key accumulator observes that $n_{\ell_n+1-j} = n_{4-j} = 2^{j-1}$ for all $j \in [3]$, and sets $\delta = 3$. The key accumulator now deletes $\mathsf{EncTree}_i$ for each $i \geq \ell_n+1-\delta = 1$, and sets $(\mathsf{EncTree}'_1, n'_1) = (\mathsf{NewTree}, 2 \cdot n_1 = 8)$. So, now the updated auxilliary information is $\mathsf{aux} = \{\mathsf{IDTree}, (\mathsf{EncTree}'_1, 8) : [\mathsf{id}_1, \ldots, \mathsf{id}_8]\}$.

Figure 9.1: An example demonstrating aux being updated during registration

For $i \in [\ell_n]$:

— For $j \in [d_i]$: It constructs a step-circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as defined in Figure 9.2 with $\mathsf{hk}, \mathsf{id}, m$, $\mathsf{state}_{i,j+1}$, $r_{i,j+1}$ hardwired. It then garbles the circuit as $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{i,j}, \mathsf{state}_{i,j})$.

— It computes the hash value of root node as $h_i = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{rt}_i)$, and computes the input garbling as $\widetilde{y}_{i,1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, h_i, \mathsf{state}_{i,1}; r_{i,1})$.

3. Finally, it outputs the ciphertext ct as $\mathsf{ct} = \Big\{ \{(\mathsf{rt}_i, d_i)\}_i, \Big\{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\Big\}_{i,j}, \{\widetilde{y}_{i,1}\}_i \Big\}$.

Circuit $\mathsf{Enc\text{-}Step}_{i,j}$

Constants: $\mathsf{hk}, \mathsf{id}, m, \mathsf{state}_{i,j+1}, r_{i,j+1}$.
Input: $\mathsf{flag}||a||\mathsf{id}^*||b \in \{0,1\}^{3\lambda+1}$.

1. If $\mathsf{flag} = 1$ and $\mathsf{id}^* = \mathsf{id}$, output $1||\mathsf{PKE.Enc}(b, m; r_{i,j+1})$.
2. If $\mathsf{flag} = 1$ and $\mathsf{id}^* \neq \mathsf{id}$, output $1||\perp$.
3. If $\mathsf{id} > \mathsf{id}^*$, output $0||\mathsf{HG.GarbleInp}(\mathsf{hk}, b, \mathsf{state}_{i,j+1}; r_{i,j+1})$
   Else, output $0||\mathsf{HG.GarbleInp}(\mathsf{hk}, a, \mathsf{state}_{i,j+1}; r_{i,j+1})$.

Figure 9.2: Description of the step-circuit $\mathsf{Enc\text{-}Step}_{i,j}$

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to u$. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and $\mathsf{aux} = \left(\mathsf{IDTree}, \{(\mathsf{EncTree}_i, n_i)\}_{i\in[\ell_n]}\right)$.

The update computation is a two-step approach. In the first step, the algorithm performs a binary search over the IDTree to obtain the timestamp associated with the identity id. As IDTree is a balanced binary search tree, this can be done efficiently. Let t be the timestamp associated with id that the binary search outputs. (It aborts if no such timestamp exists.)

In the second phase, the update generator computes the index $i^* \in [\ell_n]$ such that $\sum_{j\in[i^*-1]} n_j < t \leq \sum_{j\in[i^*]} n_j$. Index i^* corresponds to the smallest index of the encryption tree in which id has been registered. Now the algorithm performs a binary search for identity id in the encryption tree $\mathsf{EncTree}_{i^*}$. It stores the path of nodes traversed from root rt_{i^*} to leaf node containing identity id. Let path be the searched path in tree $\mathsf{EncTree}_{i^*}$. Finally, it outputs the update u as $u = \mathsf{path}$. (Again, it aborts if no such index or a path to a leaf node containing identity id exists.)

$\mathsf{Dec}(\mathsf{sk}, u, \mathsf{ct}) \to m/\perp/\mathsf{GetUpd}$. The decryption algorithm first parses the inputs as:

$$\mathsf{ct} = \left(\{(\mathsf{rt}_i, d_i)\}_i, \left\{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\right\}_{i,j}, \{\widetilde{y}_{i,1}\}_i \right), \text{ and}$$

$$u = \mathsf{path} = (\mathsf{node}_1, \ldots, \mathsf{node}_{d-1}, \mathsf{node}_d).$$

It then proceeds as follows:

1. Let i be the smallest index $i \in [\ell_n]$ such that $\mathsf{node}_1 = \mathsf{rt}_i$. If such an i does not exist, then it outputs GetUpd. Otherwise, it continues.
2. Now the decryptor iteratively runs the hash garbling evaluation algorithms as follows.

   For $j \in [d_i]$:

   — It evaluates the j^{th} step-circuit as $(\mathsf{flag}||\widetilde{y}_{i,j+1}) \leftarrow \mathsf{HG.Eval}(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \widetilde{y}_{i,j}, \mathsf{node}_i)$.

   — If $\mathsf{flag} = 1$ and $\widetilde{y}_{i,j+1} = \perp$, the algorithm outputs $\perp$.

   — Otherwise, if $\mathsf{flag} = 1$ and $\widetilde{y}_{i,j+1} \neq \perp$, then interpret $\widetilde{y}_{i,j+1}$ as a PKE ciphertext, and decrypt it as $\widetilde{y}_{i,j+1}$ using key sk to obtain the message as $m \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \widetilde{y}_{i,j+1})$. And, it outputs the message m.
3. If the algorithm did not terminate, then it outputs $\perp$.

$\mathsf{PreProve}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \pi$. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i \in [\ell_n]}$ and $\mathsf{aux} = \left(\mathsf{IDTree}, \{(\mathsf{EncTree}_i, n_i)\}_{i \in [\ell_n]}\right)$. The pre-registration proof consists of ℓ_n

sub-proofs π_i for $i \in [\ell_n]$, where each sub-proof π_i consist of two[4] adjacent paths in the i^{th} encryption tree $\mathsf{EncTree}_i$. Concretely, the algorithm proceeds as follows:

For $i \in [\ell_n]$:

— It runs a binary search on tree $\mathsf{EncTree}_i$ to find identity id. If id is contained in $\mathsf{EncTree}_i$, then it outputs $\perp$. Otherwise, it continues.

— It runs an extended binary search on tree $\mathsf{EncTree}_i$ to find two adjacent paths $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ for identities $\mathsf{id}_{i,\mathsf{lwr}}$ and $\mathsf{id}_{i,\mathsf{upr}}$, respectively. (Here $\mathsf{id}_{i,\mathsf{lwr}}$ is the largest identity in $\mathsf{EncTree}_i$ such that $\mathsf{id}_{i,\mathsf{lwr}} < \mathsf{id}$ and similarly $\mathsf{id}_{i,\mathsf{upr}}$ is the smallest identity in $\mathsf{EncTree}_i$ such that $\mathsf{id}_{i,\mathsf{upr}} > \mathsf{id}$.)

If $\mathsf{id}_{i,\mathsf{lwr}}$ is the largest identity registered in the tree $\mathsf{EncTree}_i$, that is no such $\mathsf{id}_{i,\mathsf{upr}}$ exists, then path $\mathsf{path}_{i,\mathsf{upr}}$ is set as $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$. Similarly, if $\mathsf{id}_{i,\mathsf{upr}}$ is the smallest identity, that is no such $\mathsf{id}_{i,\mathsf{lwr}}$ exists, then path $\mathsf{path}_{i,\mathsf{lwr}}$ is set as $\mathsf{path}_{i,\mathsf{lwr}} = \epsilon$.

— It sets sub-proof π_i as $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$.

Finally, it outputs the pre-registration proof as $\pi = (\pi_1, \ldots, \pi_{\ell_n})$.

$\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) \to 0/1$. Let $\mathsf{crs} = \mathsf{hk}$, $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i \in [\ell_n]}$, $\pi = (\pi_i)_{i \in [\ell_n]}$.[5]

[4]Sometimes one of the paths might just be an empty path.

[5]If the number of sub-proofs and number of encryption trees are distinct, then the verifier rejects. Here we simply consider that while parsing the inputs, the verifier verifies that the crs and pp are consistent which simply corresponds to checking that the number of trees and their depths are consistent.

Also, let each sub-proof be $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$ for $i \in [\ell_n]$.

The pre-registration proof-verification procedure proceeds as follows. For every $i \in [\ell_n]$, it runs the pre-registration sub-proof verification procedure which is described in Figures 9.3 and 9.4.

If the pre-registration sub-proof verification procedure rejects any index $i \in [\ell_n]$, then the main verification algorithm also rejects and outputs 0. Otherwise, if all sub-proof verification routines accept, then the main verification algorithm also accepts and outputs 1.

$\mathsf{PostProve}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \pi$. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and $\mathsf{aux} = \Big(\mathsf{IDTree}, \{(\mathsf{EncTree}_i, n_i)\}_{i\in[\ell_n]}\Big)$. The post-registration proof consists of ℓ_n sub-proofs π_i for $i \in [\ell_n]$, where each sub-proof π_i consist of either two or *three* adjacent paths in the i^{th} encryption tree $\mathsf{EncTree}_i$.[6] (Very briefly, having 3 adjacent paths w.r.t. an encryption tree will correspond to the proof of uniqueness of decryptability by the registered user's secret key; whereas 2 adjacent paths will mostly correspond to a proof of non-decryptability.) Concretely, the algorithm proceeds as follows:

Initialize $\ell = \perp$, where ℓ will eventually denote the index of the first encryption tree $\mathsf{EncTree}_\ell$ in which identity id was registered. For $i \in [\ell_n]$:

— It runs a binary search on tree $\mathsf{EncTree}_i$ to find identity id. If the tree contains a leaf node of the form $1||0^\lambda||\mathsf{id}||\mathsf{pk}'$ for some key $\mathsf{pk}' \neq \mathsf{pk}$,

[6] Sometimes one of the paths might just be an empty path.

Verification procedure for pre-registration sub-proof

For simplicity of exposition, suppose that none of paths $\mathsf{path}_{i,\mathsf{lwr}}$, $\mathsf{path}_{i,\mathsf{upr}}$ are empty. Towards the end, we explain how the verification handles the case if either of these paths is ϵ.

Non-empty paths. It interprets every path $\mathsf{path}_{i,\mathsf{tag}}$ as $(\mathsf{node}_{i,1,\mathsf{tag}}, \ldots, \mathsf{node}_{i,d_i,\mathsf{tag}})$ for $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$. And every node $\mathsf{node}_{i,j,\mathsf{tag}}$, is interpreted as $(\mathsf{flag}_{i,j,\mathsf{tag}}||a_{i,j,\mathsf{tag}}||\mathsf{id}_{i,j,\mathsf{tag}}||b_{i,j,\mathsf{tag}})$.

1. First, it checks that both paths $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ are well-formed. That is, $\mathsf{node}_{i,1,\mathsf{tag}} = \mathsf{rt}_i$ for both $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$. Also, it checks that $\mathsf{node}_{i,j+1,\mathsf{tag}}$ is either left child of $\mathsf{node}_{i,j,\mathsf{tag}}$ (i.e., $a_{i,j,\mathsf{tag}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}})$ and $\mathsf{id}_{i,j,\mathsf{tag}} \geq \mathsf{id}_{i,j+1,\mathsf{tag}}$), or right child of $\mathsf{node}_{i,j,\mathsf{tag}}$ (i.e., $b_{i,j,\mathsf{tag}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}})$ and $\mathsf{id}_{i,j,\mathsf{tag}} < \mathsf{id}_{i,j+1,\mathsf{tag}}$). If $\mathsf{node}_{i,j+1,\mathsf{tag}}$ is left child of $\mathsf{node}_{i,j,\mathsf{tag}}$, then it checks that $\mathsf{id}_{i,k,\mathsf{tag}} \leq \mathsf{id}_{i,j,\mathsf{tag}}$ for each $k > j$. Similarly, If $\mathsf{node}_{i,j+1,\mathsf{tag}}$ is right child of $\mathsf{node}_{i,j,\mathsf{tag}}$, then it checks that $\mathsf{id}_{i,k,\mathsf{tag}} > \mathsf{id}_{i,j,\mathsf{tag}}$ for each $k > j$. And, it checks that $\mathsf{flag}_{i,j,\mathsf{tag}} = 0$ for $j < d_i$, and $\mathsf{flag}_{i,d_i,\mathsf{tag}} = 1$, $a_{i,d_i,\mathsf{tag}} = 0^\lambda$. (Note that during this validity check, the verifier also stores whether that node is left child or right child.).
2. Next, it checks that $\mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id} < \mathsf{id}_{i,d_i,\mathsf{upr}}$, that is the identity in the *lower* path is less than that in the *upper* path, and the identity id whose non-registration is being proven lies between both these identities.

Figure 9.3: Conditions for verifying a proof $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$ that id is NOT registered as per $\mathsf{EncTree}_i$

then the algorithm simply outputs $\perp$. Otherwise, it continues as follows.

— If id is *not* contained in $\mathsf{EncTree}_i$, then it first checks that $\ell = \perp$. If the check fails, it aborts. Otherwise, it proceeds as for the pre-registration sub-proof which is to run an extended binary search on tree $\mathsf{EncTree}_i$ to find two adjacent paths $\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}}$ for identities $\mathsf{id}_{i,\mathsf{lwr}}, \mathsf{id}_{i,\mathsf{upr}}$ (respectively). Here $\mathsf{id}_{i,\mathsf{lwr}}$ is the largest identity

Verification procedure for pre-registration sub-proof (Cont'd)

3. It then computes the largest common prefix of nodes in paths $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$. That is, let k be the largest index such that $\mathsf{node}_{i,j,\mathsf{lwr}} = \mathsf{node}_{i,j,\mathsf{upr}}$ for all $j \leq k$. It checks that $\mathsf{id}_{i,k,\mathsf{lwr}} = \mathsf{id}_{i,d_i,\mathsf{lwr}}$. Also, it checks:

   (a) It checks that $\mathsf{node}_{i,k+1,\mathsf{lwr}}$ and $\mathsf{node}_{i,k+1,\mathsf{upr}}$ are *left* and *right* children of $\mathsf{node}_{i,k,\mathsf{lwr}} = \mathsf{node}_{i,k,\mathsf{upr}}$. That is, $a_{i,k,\mathsf{lwr}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,k+1,\mathsf{lwr}})$ and $b_{i,k,\mathsf{upr}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,k+1,\mathsf{upr}})$.

   (b) For every index $j > k$, $\mathsf{node}_{i,j+1,\mathsf{lwr}}$ and $\mathsf{node}_{i,j+1,\mathsf{upr}}$ are *right* and *left* children of $\mathsf{node}_{i,j,\mathsf{lwr}}$ and $\mathsf{node}_{i,j,\mathsf{upr}}$, respectively. That is, $b_{i,j,\mathsf{lwr}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{lwr}})$ and $a_{i,j,\mathsf{upr}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{upr}})$.

It rejects, i.e. outputs 0, if any of these checks fails. Otherwise, it accepts and outputs 1.

One empty path. Suppose $\mathsf{path}_{i,\mathsf{lwr}} = \epsilon$. The verifier checks first well-formedness of $\mathsf{path}_{i,\mathsf{upr}}$ as in Step 1 (above). Next, it checks that $\mathsf{id} < \mathsf{id}_{i,d_i,\mathsf{upr}}$, and lastly verifies that $\mathsf{id}_{i,d_i,\mathsf{upr}}$ is the smallest registered node in $\mathsf{EncTree}_i$. For the last check, the verifier check that for every index j, $\mathsf{node}_{i,j+1,\mathsf{upr}}$ is the *left* child of $\mathsf{node}_{i,j,\mathsf{upr}}$. It rejects, i.e., outputs 0, if any of these checks fail. Otherwise, it accepts and outputs 1.

Similarly, if $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$, then it proceeds as above, except it checks that $\mathsf{id}_{i,d_i,\mathsf{lwr}}$ is the largest identity in $\mathsf{EncTree}_i$ instead.

Figure 9.4: Conditions for verifying a proof $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$ that id is NOT registered as per $\mathsf{EncTree}_i$ (Cont'd)

in $\mathsf{EncTree}_i$ such that $\mathsf{id}_{i,\mathsf{lwr}} < \mathsf{id}$ and similarly $\mathsf{id}_{i,\mathsf{upr}}$ is the smallest identity in $\mathsf{EncTree}_i$ such that $\mathsf{id}_{i,\mathsf{upr}} > \mathsf{id}$. And, it sets sub-proof π_i as $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. (Recall that one of these paths might be empty.)

— If id is contained in $\mathsf{EncTree}_i$, then it proceeds as follows:

— If $\ell = \perp$, then it sets $\ell = i$ (i.e., sets ℓ as the first tree where id was found).

— It runs an extended binary search on tree $\mathsf{EncTree}_i$ to find three adjacent paths $\mathsf{path}_{i,\mathsf{lwr}}$, $\mathsf{path}_{i,\mathsf{mid}}$, $\mathsf{path}_{i,\mathsf{upr}}$ for identities $\mathsf{id}_{i,\mathsf{lwr}}$, id, $\mathsf{id}_{i,\mathsf{upr}}$ (respectively). Here $\mathsf{id}_{i,\mathsf{lwr}}$ is the largest identity in $\mathsf{EncTree}_i$ such that $\mathsf{id}_{i,\mathsf{lwr}} < \mathsf{id}$ and similarly $\mathsf{id}_{i,\mathsf{upr}}$ is the smallest identity in $\mathsf{EncTree}_i$ such that $\mathsf{id}_{i,\mathsf{upr}} > \mathsf{id}$.

If id is the largest identity registered in the tree $\mathsf{EncTree}_i$, that is no such $\mathsf{id}_{i,\mathsf{upr}}$ exists, then path $\mathsf{path}_{i,\mathsf{upr}}$ is set as $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$. Similarly, if id is the smallest identity, that is no such $\mathsf{id}_{i,\mathsf{lwr}}$ exists, then path $\mathsf{path}_{i,\mathsf{lwr}}$ is set as $\mathsf{path}_{i,\mathsf{lwr}} = \epsilon$.

— It sets sub-proof π_i as $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$.

Finally, it outputs the post-registration proof as $\pi = (\pi_1, \ldots, \pi_{\ell_n}, \ell)$. (Note that the cut-off index ℓ in included as part of the proof.)

$\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}, \pi) \to 0/1$. Let $\mathsf{crs} = \mathsf{hk}$, $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i \in [\ell_n]}$, $\pi = (\pi_1, \ldots, \pi_{\ell_n}, \ell)$.[7] Now each sub-proof either is interpreted as 3 adjacent paths $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$, or as 2 adjacent paths $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$ for every i.

The post-registration proof verification procedure proceeds as follows.

[7] If the number of sub-proofs and number of encryption trees are distinct, then the verifier rejects. Here we simply consider that while parsing the inputs, the verifier verifies that the crs and pp are consistent which simply corresponds to checking that the number of trees and their depths are consistent.

For every $i \in [\ell]$, it runs the *pre-registration* sub-proof verification procedure which is described in Figures 9.3 and 9.4. Now, for every $i \in \{\ell, \ell+1, \ldots, \ell_n\}$, it runs the *post-registration* sub-proof verification procedure which is described in Figures 9.5 and 9.6.

If any of the pre-registration or post-registration sub-proof verification procedures reject any index $i \in [\ell_n]$, the main verification algorithm also rejects and outputs 0. Otherwise, if all sub-proof verification routines accept, then the main verification algorithm also accepts and outputs 1.

Remark 9.1.1. *In the above construction, we make the key accumulator maintain a special balanced tree* IDTree *privately. It turns out this is unnecessary. One could easily remove it from our construction, thereby only leaving the list of encryption trees* $\{\mathsf{EncTree}_i\}_i$ *as part of the auxiliary information. However, we include* IDTree *explicitly as part of the description for ease of exposition.*

9.2 Efficiency

In this section, we prove that our scheme satisfies the compactness and efficiency requirements of Definitions 8.3.1 and 8.3.2. The analysis of Reg and Upd algorithms is similar to the analysis presented in [48].

Time Complexity of Reg Algorithm. Given an identity-key pair $(\mathsf{id}, \mathsf{pk})$, the registration algorithm first updates its timestep counter $n := n + 1$, and inserts (id, n) tuple into the balanced binary tree IDTree, which is sorted as per identities. This insertion takes $O(\log n)$ time.

Verification procedure for post-registration sub-proof

For simplicity of exposition, suppose that none of paths $\mathsf{path}_{i,\mathsf{lwr}}$, $\mathsf{path}_{i,\mathsf{upr}}$ are empty. Towards the end, we explain how the verification handles the case if either of these paths is ϵ.

Non-empty paths. It interprets every path $\mathsf{path}_{i,\mathsf{tag}}$ as $(\mathsf{node}_{i,1\mathsf{tag}}, \ldots, \mathsf{node}_{i,d_i\mathsf{tag}})$ for $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$. And every node $\mathsf{node}_{i,j,\mathsf{tag}}$, is interpreted as $(\mathsf{flag}_{i,j,\mathsf{tag}}||a_{i,j,\mathsf{tag}}||\mathsf{id}_{i,j,\mathsf{tag}}||b_{i,j,\mathsf{tag}})$.

1. First, it checks that both paths $\mathsf{path}_{i,\mathsf{lwr}}$, $\mathsf{path}_{i,\mathsf{mid}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ are well-formed. That is, $\mathsf{node}_{i,1,\mathsf{tag}} = \mathsf{rt}_i$ for both $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$. Also, it checks that $\mathsf{node}_{i,j+1,\mathsf{tag}}$ is either a left child of $\mathsf{node}_{i,j,\mathsf{tag}}$ (i.e., $a_{i,j,\mathsf{tag}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}})$ and $\mathsf{id}_{i,j,\mathsf{tag}} \geq \mathsf{id}_{i,j+1,\mathsf{tag}}$), or is a right child of $\mathsf{node}_{i,j,\mathsf{tag}}$ (i.e., $b_{i,j,\mathsf{tag}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}})$ and $\mathsf{id}_{i,j,\mathsf{tag}} < \mathsf{id}_{i,j+1,\mathsf{tag}}$). If $\mathsf{node}_{i,j+1,\mathsf{tag}}$ is left child of $\mathsf{node}_{i,j,\mathsf{tag}}$, then it checks that $\mathsf{id}_{i,k,\mathsf{tag}} \leq \mathsf{id}_{i,j,\mathsf{tag}}$ for each $k > j$. Similarly, If $\mathsf{node}_{i,j+1,\mathsf{tag}}$ is right child of $\mathsf{node}_{i,j,\mathsf{tag}}$, then it checks that $\mathsf{id}_{i,k,\mathsf{tag}} > \mathsf{id}_{i,j,\mathsf{tag}}$ for each $k > j$. And, it checks that $\mathsf{flag}_{i,j,\mathsf{tag}} = 0$ for $j < d_i$, and $\mathsf{flag}_{i,d_i,\mathsf{tag}} = 1$, $a_{i,d_i,\mathsf{tag}} = 0^\lambda$. (Note that during this validity check, the verifier also stores whether that node is left child or right child.)

2. Next, it checks that $\mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id} = \mathsf{id}_{i,d_i,\mathsf{mid}} < \mathsf{id}_{i,d_i,\mathsf{upr}}$, that is the identity in the *lower* path is less than that in the *upper* path, and the identity id whose non-registration is being proven is equal to the identity in the *middle* path and lies between the other two identities. It also checks that $b_{i,d_i,\mathsf{mid}} = \mathsf{pk}$.

Figure 9.5: Conditions for verifying a proof $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$ that id is registered as per $\mathsf{EncTree}_i$

The algorithm then inserts $(\mathsf{id}, \mathsf{pk})$ tuple into a balanced binary tree $\mathsf{EncTree}_{\ell_n}$. This involves performing binary search on id in $\mathsf{EncTree}_{\ell_n}$ to identify the leaf at which $1||0^\lambda||\mathsf{id}||\mathsf{pk}$ is to be inserted. As the tree is balanced, the binary search takes $O(\log n)$ time. The registration algorithm then inserts the leaf and balances the tree as per the rules of the underlying data structure. This step also takes $O(\log n)$ time. As inserting leaf and balancing the

Verification procedure for post-registration sub-proof (Cont'd)

3. For both tag pairs $(\mathsf{tag}_1, \mathsf{tag}_2) \in \{(\mathsf{lwr}, \mathsf{mid}), (\mathsf{mid}, \mathsf{upr})\}$, it proceeds as follows:

   It computes the largest common prefix of nodes in paths $\mathsf{path}_{i,\mathsf{tag}_1}$ and $\mathsf{path}_{i,\mathsf{tag}_2}$. That is, let k be the largest index such that $\mathsf{node}_{i,j,\mathsf{tag}_1} = \mathsf{node}_{i,j,\mathsf{tag}_2}$ for all $j \leq k$. It checks that $\mathsf{id}_{i,k,\mathsf{tag}_1} = \mathsf{id}_{i,d_i,\mathsf{tag}_1}$. Also, it checks:

   (a) It checks that $\mathsf{node}_{i,k+1,\mathsf{tag}_1}$ and $\mathsf{node}_{i,k+1,\mathsf{tag}_2}$ are *left* and *right* children of $\mathsf{node}_{i,k,\mathsf{tag}_1} = \mathsf{node}_{i,k,\mathsf{tag}_2}$. That is, $a_{i,k,\mathsf{tag}_1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,k+1,\mathsf{tag}_1})$ and $b_{i,k,\mathsf{tag}_2} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,k+1,\mathsf{tag}_2})$.

   (b) For every index $j > k$, $\mathsf{node}_{i,j+1,\mathsf{tag}_1}$ and $\mathsf{node}_{i,j+1,\mathsf{tag}_2}$ are *right* and *left* children of $\mathsf{node}_{i,j,\mathsf{tag}_1}$ and $\mathsf{node}_{i,j,\mathsf{tag}_2}$, respectively. That is, $b_{i,j,\mathsf{tag}_1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}_1})$ and $a_{i,j,\mathsf{tag}_2} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}_2})$.

It rejects, i.e. outputs 0, if any of these checks fails. Otherwise, it accepts and outputs 1.

One empty path. Suppose $\mathsf{path}_{i,\mathsf{lwr}} = \epsilon$. The verifier checks first well-formedness of $\mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}}$ as in Step 1 (above). Next, it checks that $\mathsf{id} = \mathsf{id}_{i,d_i,\mathsf{mid}} < \mathsf{id}_{i,d_i,\mathsf{upr}}$ as in Step 2 (above). And lastly, it performs the Step 3 verification checks as described above only for the tag pair $(\mathsf{tag}_1, \mathsf{tag}_2) = (\mathsf{mid}, \mathsf{upr})$. Lastly verifies that $\mathsf{node}_{i,d_i,\mathsf{mid}}$ is the smallest registered node in $\mathsf{EncTree}_i$ i.e., the verifier checks that for every index j, $\mathsf{node}_{i,j+1,\mathsf{mid}}$ is the *left* child of $\mathsf{node}_{i,j,\mathsf{mid}}$. It rejects, i.e., outputs 0, if any of these checks fail. Otherwise, it accepts and outputs 1.

Similarly, if $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$, then it proceeds complementarily to above which is to check well-formedness of $\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}$, range check $\mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id} = \mathsf{id}_{i,d_i,\mathsf{mid}}$, and lastly it performs the Step 3 verification checks only for the tag pair $(\mathsf{tag}_1, \mathsf{tag}_2) = (\mathsf{lwr}, \mathsf{mid})$. Lastly verifies that $\mathsf{node}_{i,d_i,\mathsf{mid}}$ is the largest registered node in $\mathsf{EncTree}_i$ i.e., the verifier checks that for every index j, $\mathsf{node}_{i,j+1,\mathsf{mid}}$ is the *right* child of $\mathsf{node}_{i,j,\mathsf{mid}}$. It rejects, i.e., outputs 0, if any of these checks fail. Otherwise, it accepts and outputs 1.

Figure 9.6: Conditions for verifying a proof $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$ that id is registered as per $\mathsf{EncTree}_i$ (Cont'd)

tree takes $O(\log n)$ time, at most $O(\log n)$ node labels can change during the process. For all such modified nodes, the hash values stored in their ancestors are updated. As the tree's depth is $O(\log n)$, this involves updating labels of $O(\log^2 n)$ nodes. [8]

The registration algorithm then computes δ and deletes some old versions of encryption trees. In the construction, the key accumulator maintains an invariant that the number of identities n_i associated with various trees $\mathsf{EncTree}_i$ in auxiliary information is a different power of 2. As there are n registered identities, there can be at most $\log n$ trees in auxiliary information. Therefore, computing δ and merging trees takes only $O(\log n)$ time. Overall, the registration process takes $O(\log^2 n)$ time.

Size of Public Parameters. The public parameters consist of each tree's root and depth in auxiliary information. There can be at most $\log n$ trees in auxiliary information as described earlier. Therefore, this size of public parameters is at most $O(\log n)$.

Time Complexity of Upd Algorithm. During the update process, the Upd algorithm first computes the timestep t_{id} at which the given identity id is registered by performing a binary search through a balanced binary tree

[8] Note that the registration algorithm needs to first make a copy of $\mathsf{EncTree}_{\ell_n}$, then insert the leaf $1||0^\lambda||\mathsf{id}||\mathsf{pk}$ into the copied tree to create $\mathsf{NewTree}$. Copying a tree naively takes $O(n)$ time. However, this overhead can be easily avoided by not storing the entire $\mathsf{NewTree}$. Basically, $\mathsf{NewTree}$ only stores values of the $O(\log^2 n)$ nodes that are different from $\mathsf{EncTree}_{\ell_n}$.

IDTree. As IDTree is balanced, the binary search requires only $O(\log n)$ time. After computing t_{id}, the Upd algorithm identifies the balanced binary tree $\mathsf{EncTree}_i$ that is associated with timestep t_{id}. As described earlier, there can be at most $\log n$ trees, and therefore this operation takes $O(\log n)$ time. The key accumulator then performs a binary search on id in the associated tree $\mathsf{EncTree}_i$ and outputs the sequence of nodes traversed during the binary search. As $\mathsf{EncTree}_i$ is balanced, this step also requires $O(\log n)$ time. Overall, the update algorithm runs in $O(\log n)$ time.

Size of an Update. The Upd algorithm performs a binary search on the given identity id in a balanced binary tree $\mathsf{EncTree}_i$, which contains registered identity-key pairs. The algorithm then outputs the sequence of nodes traversed during the binary search. As the tree is balanced and as there are at most n identities in the tree, the depth of the tree is $O(\log n)$. Therefore, the size of an update output by Upd algorithm is $O(\log n)$.

Number of Updates. In our construction, we say that a registered identity id with timestamp t is associated with $\mathsf{EncTree}_j$ if $\sum_{k<j} n_k < t \leq \sum_{k\leq j} n_k$. A registered identity id needs to invoke Upd algorithm only when the tree $\mathsf{EncTree}_j$ associated with id gets deleted during Reg algorithm. By our construction, any pair $(\mathsf{EncTree}_i, n_i)$ gets deleted only when n_i new identites are registered after $(\mathsf{EncTree}_i, n_i)$ is added to aux. Therefore for any identity id, its i^{th} update happens when 2^i new identities are registered after its $(i-1)^{th}$

update. As there are at most n registrations, the number of updates received by any identity is at most $\log n$.

Time Complexity of PreProve/PostProve **Algorithms.** Given an identity id, the PreProve/PostProve algorithms perform binary search on at most 3 identities in each balanced binary tree $\mathsf{EncTree}_i$ present in the aux. As each tree in aux is balanced, contains at most n leaves, and is ordered as per identities, a binary search on each tree takes $O(\log n)$ time. In the construction, the key accumulator maintains an invariant that the number of identities n_i associated with various trees $\mathsf{EncTree}_i$ in auxiliary information is a different power of 2. As there are n registered identities, there can be at most $\log n$ trees in auxiliary information. Therefore, the time complexity of PreProve and PostProve algorithms is at most $O(\log^2 n)$.

Size of Pre/Post-Registration Proofs. The pre/post-registration subproofs consist of at most 3 paths (sequence of nodes from root to a leaf) for each tree in auxiliary information. As each tree in aux is balanced and contains at most n leaves, each tree's depth is $O(\log n)$. In the construction, the key accumulator maintains an invariant that the number of identities n_i associated with various trees $\mathsf{EncTree}_i$ in auxiliary information is a different power of 2. As there are n registered identities, there can be at most $\log n$ trees in auxiliary information. Therefore, the size of pre/post-registration proofs is $O(\log^2 n)$.

9.3 Completeness

We now show that the above scheme satisfies completenesss requirements described in Definitions 8.3.1 to 8.3.3.

9.3.1 Completeness of VRBE

Lemma 9.3.1. *Assuming* HG *and* PKE *are perfectly correct, the above construction satisfies completeness property (Definition 8.3.1)*[9].

Proof. Consider any computationally unbounded adversary $\mathcal{A}$. The challenger first initializes $(\mathsf{pp}, \mathsf{aux}, u, S_{\mathsf{ID}}, \mathsf{id}^*) = (\epsilon, \epsilon, \epsilon, \emptyset, \perp)$, samples $\mathsf{crs} = \mathsf{hk}$ and sends it to $\mathcal{A}$. The adversary requests for polynomially many registration queries of the form $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$ or $(\mathsf{regtgt}, \mathsf{id}^*)$. For each $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$ query, the challenger registers $(\mathsf{id}, \mathsf{pk})$. For $(\mathsf{regtgt}, \mathsf{id}^*)$ query, the challenger samples PKE pair $(\mathsf{pk}^*, \mathsf{sk}^*)$, registers $(\mathsf{id}^*, \mathsf{pk}^*)$ and sends back pk^* to the adversary. The adversary then makes an encryption query (enctgt, m) and obtains
$\mathsf{ct} = (\mathsf{pp}, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}, \{\tilde{y}\}_{i,1}) \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{id}^*, m)$, where id^* is the challenge identity and $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_i$ is public parameters at the time of encryption query. The adversary then makes few more registration queries and then requests to decrypt the ciphertext ct. The challenger then runs the decryption algorithm $x = \mathsf{Dec}(\mathsf{sk}^*, u, \mathsf{ct})$. We know that x is either GetUpd

[9]For the sake of simplicity, we consider the completeness game in which the adversary makes only one encryption and decryption query. Note that if an adversary can break completeness property with access to polynomially many encryption and decryption queries (as in Definition 8.3.1), it can also break the completeness property with access to single encryption and decryption query.

or $\perp$ or a message. If $x = \mathsf{GetUpd}$, the challenger runs the update algorithm $u \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ and then again runs decryption algorithm $x = \mathsf{Dec}(\mathsf{sk}^*, u, \mathsf{ct})$. Assuming $x \neq \mathsf{GetUpd}$, we now prove that x is always equals to m. Let u be $(\mathsf{node}_1, \mathsf{node}_2, \cdots \mathsf{node}_d)$ and $\mathsf{node}_i = \mathsf{flag}_i||a_i||\mathsf{id}_i||b_i$ for each i. We first make the following observations. (1) $\mathsf{flag}_i = 1$ iff $i = d$, (2) $\mathsf{id}_d = \mathsf{id}^*$, $b_d = \mathsf{pk}^*$, (3) $\mathsf{node}_1 = \mathsf{rt}_\kappa$ for some $\kappa \in [\ell_n]$, and (4) for each $i < d$, $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i+1}) = \begin{cases} a_i & \text{if } \mathsf{id}^* \leq \mathsf{id}_i \\ b_i & \text{Otherwise} \end{cases}$. The last observation is due to the fact that u is obtained by performing binary search on id^* in an $\mathsf{EncTree}$. Based on the above observations, the decryption algorithm runs $\mathsf{HEval}(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,1}, \tilde{y}_{\kappa,1}, \mathsf{node}_1)$. By the perfect correctness of hash garbling scheme, it obtains $0||\tilde{y}_{\kappa,2} = 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_2), \mathsf{state}_{\kappa,2}; r_{\kappa,2})$. The algorithm then runs $\mathsf{HEval}(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,2}, \tilde{y}_{\kappa,2}, \mathsf{node}_2)$ and obtains $0||\mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_3), \mathsf{state}_{\kappa,3}; r_{\kappa,3})$ by perfect correctness of hash garbling scheme. On continuing the process, the final circuit outputs $1||\mathsf{Enc}(\mathsf{pk}^*, m)$. The decryption algorithm then decrypts the PKE ciphertext $\mathsf{Enc}(\mathsf{pk}^*, m)$ and obtains m by perfect correctness of PKE scheme. ■

Remark 9.3.1 (Handling missing updates)**.** *Note that the above completeness proof relies on the fact that the user has the update u associated w.r.t. the public parameters* pp *used during encryption. However, this does not capture the scene when the challenger registers many identities after* ct *is created. Due to delete operations in the* Reg *algorithm, the* $\mathsf{EncTree}$ *associated with identity* id *is changed. That is, in this case, it might have happened that even after the* Upd *algorithm is executed, the root value of the update u may not match*

with any root value rt_i *used while creating the ciphertext. Thus, the decryption algorithm outputs* GetUpd *again, and therefore it is impossible to decrypt such a well-formed ciphertext. Intuitively, the issue is that the updates could get* lost *during the registration process. Identical issues were observed in the prior works [47, 48] as well. Here we sketch a simple and efficient mechanism to resolve the issue. We ignore this detail in our scheme's formal description for ease of exposition.*

The idea is to store a list of 'old' updates for every user in the IDTree *that we already maintain. To preserve efficiency, an update is added to the list if it is not already present, i.e., only after an existing encryption tree is removed. More formally, the tree* IDTree *now stores tuples of the form* $(\mathsf{id}, (t, L))$*, where* t *denotes the timestamp as before, and* L *denotes the list of all the past updates (path from the root of an encryption tree to the leaf containing the identity* id*). Now during the registration, when an encryption tree* EncTree *associated with an identity* id *changes (i.e., is removed), then the key accumulator for every such identity* id *appends the update information (i.e., the path from the root to leaf containing* id*) to the tuple* $(\mathsf{id}, (t, L))$ *in the tree* IDTree*. Note that this can be efficiently done by performing a binary search on* IDTree *and does not blow up the size of public parameters. For generating updates, the algorithm searches for the given identity* id *in* IDTree *to obtain the tuple* $(\mathsf{id}, (t, L))$*. It then sends the list* L *containing all the past updates in addition to the current update information. On receiving the list of all the past updates, the decryption algorithm decrypts the ciphertext with each of the*

updates. It then outputs the message if any update works.

Lastly, such a scenario could be more easily handled in real-world applications by issuing out updates after every deletion instead of storing them. That is, when the key accumulator possibly deletes existing encryption trees (Step (3) in Reg *algorithm), then it runs the update algorithm* $\mathsf{Upd}(\mathsf{pp}, \mathsf{id})$ *and sends the update for each identity* id *with timestamp greater than* $\sum_{i<\ell_n+1-\delta} n_i$.

9.3.2 Completeness of Pre/Post-Registration Verifiability

Lemma 9.3.2. *The above construction satisfies completeness of pre/post-Registration Verifiability property (Definition 8.3.2)*[10].

Proof. Consider any computationally unbounded adversary $\mathcal{A}$. The challenger first initializes $(\mathsf{pp}, \mathsf{aux}, u, S_{\mathsf{ID}}, \mathsf{id}^*) = (\epsilon, \epsilon, \epsilon, \emptyset, \perp)$, samples $\mathsf{crs} = \mathsf{hk}$ and sends it to $\mathcal{A}$. The adversary requests for polynomially many registration queries of the form $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$. For each $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$ query, the challenger registers $(\mathsf{id}, \mathsf{pk})$ as $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$. The adversary then either queries $(\mathsf{prereg}, \mathsf{id}^*)$ on any unregistered identity id^* or $(\mathsf{postreg}, \mathsf{id}^*, \mathsf{pk}^*)$ on any registered key-pair $(\mathsf{id}^*, \mathsf{pk}^*)$.

Suppose $\mathcal{A}$ makes query $(\mathsf{prereg}, \mathsf{id}^*)$ on any unregistered identity id^*. The challenger computes $\pi = \mathsf{PreProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*)$. We argue that

[10]For the sake of simplicity, we consider the completeness game in which the adversary makes only one prereg or postreg query. Note that if an adversary can break completeness property with access to polynomially many prereg and postreg queries (as in Definition 8.3.2), it can also break the completeness property with access to single prereg or postreg query.

$\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi) = 1$ with probability 1. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i \in [\ell_n]}$ and $\pi = (\pi_1, \cdots, \pi_{\ell_n})$, where $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$ for each i. For each i and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}} = (\mathsf{node}_{i,1,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$, where $\mathsf{node}_{i,j,\mathsf{tag}} = \mathsf{flag}_{i,j,\mathsf{tag}} || a_{i,j,\mathsf{tag}} || \mathsf{id}_{i,j,\mathsf{tag}} || b_{i,j,\mathsf{tag}}$. As $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ are obtained by performing binary search in $\mathsf{EncTree}_i$, they satisfy well-formedness conditions (Point 1 in Figures 9.3 and 9.4). As id^* is an unregistered user, the extended binary search on id^* in $\mathsf{EncTree}_i$ leads to identities $\mathsf{id}_{i,d_i,\mathsf{lwr}}$ and $\mathsf{id}_{i,d_i,\mathsf{upr}}$ s.t. $\mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id}^* < \mathsf{id}_{i,d_i,\mathsf{upr}}$. As $\mathsf{EncTree}_i$ is ordered as per identites and as there are no registered identities between $\mathsf{id}_{i,d_i,\mathsf{lwr}}$ between $\mathsf{id}_{i,d_i,\mathsf{upr}}$, the paths $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ are 'adjacent' and therefore satisfies condition 3 in Figures 9.3 and 9.4.

On the other hand, suppose $\mathcal{A}$ makes query $(\mathsf{postreg}, \mathsf{id}^*, \mathsf{pk}^*)$ on any registered identity-key pair $(\mathsf{id}^*, \mathsf{pk}^*)$. The challenger computes $\pi = \mathsf{PostProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$. We argue that $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 1$ with probability 1. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i \in [\ell_n]}$ and $\pi = (\pi_1, \cdots, \pi_{\ell_n}, \alpha)$. For any $i < \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \geq \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. For each i and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}} = (\mathsf{node}_{i,1,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$, where $\mathsf{node}_{i,j,\mathsf{tag}} = \mathsf{flag}_{i,j,\mathsf{tag}} || a_{i,j,\mathsf{tag}} || \mathsf{id}_{i,j,\mathsf{tag}} || b_{i,j,\mathsf{tag}}$. As the paths are obtained by performing binary search in $\mathsf{EncTree}_i$, they satisfy well-formedness conditions (Point 1 in Figures 9.3 to 9.6). For each $i < \alpha$, we know that id^* is unregistered in $\mathsf{EncTree}_i$, and therefore extended binary search on id^* in $\mathsf{EncTree}_i$ leads to identities $\mathsf{id}_{i,d_i,\mathsf{lwr}}$ and $\mathsf{id}_{i,d_i,\mathsf{upr}}$ s.t. $\mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id}^* < \mathsf{id}_{i,d_i,\mathsf{upr}}$. Moreover, as $\mathsf{EncTree}_i$ is ordered as per identites and as there are no registered

identities between $\mathsf{id}_{i,d_i,\mathsf{lwr}}$ between $\mathsf{id}_{i,d_i,\mathsf{upr}}$, the paths $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ are 'adjacent' and therefore satisfies condition 3 in Figures 9.3 and 9.4. For each $i \geq \alpha$, we know that $(\mathsf{id}^*, \mathsf{pk}^*)$ is registered in $\mathsf{EncTree}_i$, and therefore extended binary search on id^* in $\mathsf{EncTree}_i$ leads to identities $\mathsf{id}_{i,d_i,\mathsf{lwr}}, \mathsf{id}_{i,d_i,\mathsf{mid}}, \mathsf{id}_{i,d_i,\mathsf{upr}}$ s.t. $\mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id}_{i,d_i,\mathsf{mid}} = \mathsf{id}^* < \mathsf{id}_{i,d_i,\mathsf{upr}}$. Moreover, as $\mathsf{EncTree}_i$ is ordered as per identites and as there are no registered identities between $\mathsf{id}_{i,d_i,\mathsf{lwr}}, \mathsf{id}_{i,d_i,\mathsf{mid}}$, the paths $\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}$ are 'adjacent' and therefore satisfy condition 3 in Figures 9.5 and 9.6. Similarly, the paths $\mathsf{id}_{i,d_i,\mathsf{mid}}, \mathsf{id}_{i,d_i,\mathsf{upr}}$, are 'adjacent' and satisfy condition 3 in Figures 9.5 and 9.6. ■

9.3.3 Efficiently Extractable Completeness for Post-Registration

We now prove a lemma that is useful for proving soundness of pre/post-registration properties. Consider any $\mathsf{crs} = \mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$, public parameters $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_i$, identity id^* and a sub-proof $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. Suppose, $(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi_i)$ satisfies the verification conditions described in Figures 9.3 and 9.4. We prove that if $\mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$, then $\mathsf{path}_{i,\mathsf{upr}}$ resembles a sequence of nodes obtained by performing binary search on id^* in some tree $\mathsf{EncTree}$ with root value rt_i. If $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$, we prove that $\mathsf{path}_{i,\mathsf{lwr}}$ resembles a sequence of nodes obtained by performing binary search on id^* in some tree $\mathsf{EncTree}$ with root value rt_i. Concretely, we prove the following lemma.

Lemma 9.3.3. *Consider any hash key* hk*, public parameters* $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_i$*, identity* id^* *and a sub pre-registration proof* $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$*. For any* $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$*, let* $\mathsf{path}_{i,\mathsf{tag}} = (\mathsf{node}_{i,1,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$*, where* $\mathsf{node}_{i,j,\mathsf{tag}} =$

$\mathsf{flag}_{i,j,\mathsf{tag}}||a_{i,j,\mathsf{tag}}||\mathsf{id}_{i,j,\mathsf{tag}}||b_{i,j,\mathsf{tag}}$. *If* $(\mathsf{hk}, \mathsf{pp}, \mathsf{id}^*, \pi_i)$ *satisfies the pre-registration sub-proof verification procedure of Figures 9.3 and 9.4, then* π_i *satisfies the following conditions.*

- $\mathsf{node}_{i,1,\widetilde{\mathsf{tag}}} = \mathsf{rt}_i$, $\mathsf{id}_{i,d_i,\widetilde{\mathsf{tag}}} \neq \mathsf{id}^*$, $\mathsf{flag}_{i,j,\widetilde{\mathsf{tag}}} = 1$ *iff* $j = d_i$ *and*
- *For each* $j < d_i$, $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\widetilde{\mathsf{tag}}}) = \begin{cases} a_{i,j,\widetilde{\mathsf{tag}}} & \text{if } \mathsf{id}^* \leq \mathsf{id}_{i,j,\widetilde{\mathsf{tag}}} \\ b_{i,j,\widetilde{\mathsf{tag}}} & \text{if } \mathsf{id}^* > \mathsf{id}_{i,j,\widetilde{\mathsf{tag}}} \end{cases}$

where $\widetilde{\mathsf{tag}} = \mathsf{upr}$ *and* $\mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$, *or* $\widetilde{\mathsf{tag}} = \mathsf{lwr}$ *and* $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$.

Proof. The fact that subproof π_i satisfies the first condition directly from the checks performed by PreVerify algorithm. We now prove that π_i satisfies the second condition. Let us first consider the case where $\mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$ and $\widetilde{\mathsf{tag}} = \mathsf{upr}$. We prove that if the above condition is violated for any j, then the sub-proof π_i violates one of the conditions in Figures 9.3 and 9.4.

1. Case 1 (There exists j s.t. $\mathsf{id}^* > \mathsf{id}_{i,j,\mathsf{upr}}$ and $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{upr}}) \neq b_{i,j,\mathsf{upr}}$): As π_i satisfies the well-formedness condition, we know that $\mathsf{node}_{i,j+1,\mathsf{upr}}$ is left child of $\mathsf{node}_{i,j,\mathsf{upr}}$ (i.e., $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{upr}}) = a_{i,j,\mathsf{upr}}$ and $\mathsf{id}_{i,j+1,\mathsf{upr}} \leq \mathsf{id}_{i,j,\mathsf{upr}}$). Consequently, for every $k > j$, we have $\mathsf{id}_{i,k,\mathsf{upr}} \leq \mathsf{id}_{i,j,\mathsf{upr}}$. Therefore, $\mathsf{id}_{i,d_i,\mathsf{upr}} \leq \mathsf{id}_{i,j,\mathsf{upr}} < \mathsf{id}^*$ and the sub-proof π_i violates condition 2 in Figures 9.3 and 9.4.

2. Case 2 (There exists j s.t. $\mathsf{id}^* \leq \mathsf{id}_{i,j,\mathsf{upr}}$ and $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{upr}}) \neq a_{i,j,\mathsf{upr}}$): As π_i satisfies the well-formedness condition, we know that $\mathsf{node}_{i,j+1,\mathsf{upr}}$ is right child of $\mathsf{node}_{i,j,\mathsf{upr}}$ (i.e., $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{upr}}) =$

$b_{i,j,\mathsf{upr}}$ and $\mathsf{id}_{i,j+1,\mathsf{upr}} > \mathsf{id}_{i,j,\mathsf{upr}}$). Let k be the largest index such that for all $\ell \leq k$, $\mathsf{node}_{i,\ell,\mathsf{upr}} = \mathsf{node}_{i,\ell,\mathsf{lwr}}$. We now consider 3 subcases.

- Case 2a ($j < k$): We know that $\mathsf{node}_{i,j+1,\mathsf{lwr}}$ is the right child of $\mathsf{node}_{i,j,\mathsf{lwr}}$. Consequently, by the well-formedness condition, $\mathsf{id}_{i,d_i,\mathsf{lwr}} > \mathsf{id}_{i,j,\mathsf{lwr}} \geq \mathsf{id}^*$ and the sub-proof π_i violates condition 2 in Figures 9.3 and 9.4.
- Case 2b ($j > k$): As $\mathsf{node}_{i,j+1,\mathsf{upr}}$ is right child of $\mathsf{node}_{i,j,\mathsf{upr}}$, the sub-proof π_i violates the condition (3b) in Figures 9.3 and 9.4.
- Case 2c ($j = k$): As π_i satisfies conditions 2 and 3 of Figures 9.3 and 9.4, we know that $\mathsf{id}_{i,k,\mathsf{lwr}} = \mathsf{id}_{i,d_i,\mathsf{lwr}} < \mathsf{id}^*$. This violates our assumption that $\mathsf{id}^* \leq \mathsf{id}_{i,j,\mathsf{upr}} = \mathsf{id}_{i,j,\mathsf{lwr}}$.

Let us now consider the case where $\mathsf{path}_{i,\mathsf{upr}} = \epsilon$ and $\widetilde{\mathsf{tag}} = \mathsf{lwr}$. In this case, we know that for each $j < d_i$, $\mathsf{node}_{i,j+1,\mathsf{lwr}}$ is right child of $\mathsf{node}_{i,j,\mathsf{lwr}}$ and $\mathsf{id}^* > \mathsf{id}_{i,d_i,\mathsf{lwr}}$. This implies, for each $j < d_i$, $\mathsf{id}^* > \mathsf{id}_{i,j,\mathsf{lwr}}$ and $b_{i,j,\mathsf{lwr}} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{lwr}})$. ■

We now state a similar lemma that is useful for proving efficiently extractable completeness for post-registration property and soundness of post-registration property. Consider any $\mathsf{crs} = \mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$, public parameters $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_i$, identity id^*, public key pk^* and a sub-proof $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. Suppose, $(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi_i)$ satisfies the verification conditions described in Figures 9.5 and 9.6. We prove that $\mathsf{path}_{i,\mathsf{mid}}$

resembles a sequence of nodes obtained by performing binary search on id^* in some tree $\mathsf{EncTree}$ with root value rt_i. Concretely, we prove the following lemma.

Lemma 9.3.4. *Consider any hash key* hk*, public parameters* $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_i$*, identity* id^**, public key* pk^* *and a sub-proof* $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$*. For any* $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$*, let* $\mathsf{path}_{i,\mathsf{tag}} = (\mathsf{node}_{i,1,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$*, where* $\mathsf{node}_{i,j,\mathsf{tag}} = \mathsf{flag}_{i,j,\mathsf{tag}} || a_{i,j,\mathsf{tag}} || \mathsf{id}_{i,j,\mathsf{tag}} || b_{i,j,\mathsf{tag}}$*. If* $(\mathsf{hk}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi_i)$ *satisfies the post-registration sub-proof verification procedure of Figures 9.5 and 9.6, then* π_i *satisfies the following conditions.*

- $\mathsf{node}_{i,1,\mathsf{mid}} = \mathsf{rt}_i$, $\mathsf{id}_{i,d_i,\mathsf{mid}} = \mathsf{id}^*$, $b_{i,d_i,\mathsf{mid}} = \mathsf{pk}^*$, $\mathsf{flag}_{i,j,\mathsf{mid}} = 1$ *iff* $j = d_i$ *and*
- *For each* $j < d_i$*, we have* $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{mid}}) = \begin{cases} a_{i,j,\mathsf{mid}} & \text{if } \mathsf{id}^* \leq \mathsf{id}_{i,j,\mathsf{mid}} \\ b_{i,j,\mathsf{mid}} & \text{if } \mathsf{id}^* > \mathsf{id}_{i,j,\mathsf{mid}} \end{cases}$.

Proof. This proof is similar to the proof of Lemma 9.3.3. ■

Lemma 9.3.5. *Assuming* HG *and* PKE *are perfectly correct, the above construction satisfies efficiently extractable completeness for post-registration property (Definition 8.3.3).*

Proof. We first present an UpdExt algorithm which takes a post-registration proof π and an identity id as input and outputs an update u. We then prove the construction satisfies efficiently extractable completeness for post-registration property w.r.t the UpdExt algorithm.

$\mathsf{UpdExt}(\pi, \mathsf{id}) \to \{u/\perp\}$: Let the proof π be $(\pi_1, \cdots, \pi_{\ell_n}, \alpha)$, where $\alpha \in [\ell_n]$. For each index $i < \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. For each $i \geq \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. Output $\perp$ if the proof is not in this format. Otherwise, output update $u = \mathsf{path}_{\alpha,\mathsf{mid}}$.

Consider any computationally unbounded *admissible* adversary $\mathcal{A}$. The challenger first samples $\mathsf{crs} = \mathsf{hk} \leftarrow \mathsf{CRSGen}(1^\lambda)$, a PKE key pair $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends $(\mathsf{crs}, \mathsf{pk}^*)$ to the adversary. The adversary then outputs public parameters pp, identity id^*, proof π and a message m. Let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and $\pi = (\pi_1, \cdots, \pi_{\ell_n}, \alpha)$. As $\mathcal{A}$ is admissible, we know that π is a valid post-registration proof i.e., $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 1$. The challenger first computes $\mathsf{ct} = (\mathsf{pp}, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,1}\}_{i,1}) \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m)$, extracts $u = \mathsf{UpdExt}(\pi, \mathsf{id})$ and then runs $x = \mathsf{Dec}(\mathsf{sk}, u, \mathsf{ct})$. Let $u = (\mathsf{node}_1, \cdots, \mathsf{node}_d)$, where $\mathsf{node}_i = \mathsf{flag}_i||a_i||\mathsf{id}_i||b_i$ for each i. As $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 1$, by Lemma 9.3.4, we know (1) $\mathsf{flag}_i = 1$ iff $i = d$, (2) $\mathsf{id}_d = \mathsf{id}^*$, $b_d = \mathsf{pk}^*$, (3) $\mathsf{node}_1 = \mathsf{rt}_\alpha$, and (4) for each $i < d$, $\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i+1}) = \begin{cases} a_i & \text{if } \mathsf{id}^* \leq \mathsf{id}_i \\ b_i & \text{Otherwise} \end{cases}$. Based on the above observations, we now show that $x = m$. As $\mathsf{node}_1 = \mathsf{rt}_\alpha$, the decryption algorithm runs $\mathsf{HEval}(\widetilde{\mathsf{Enc\text{-}Step}}_{\alpha,1}, \tilde{y}_{\alpha,1}, \mathsf{node}_1)$. By the perfect correctness of hash garbling scheme, this results in

$$0||\tilde{y}_{\alpha,2} = \begin{cases} 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, a_2, \mathsf{state}_{i,2}; r_{i,2}) & \text{if } \mathsf{id}^* \leq \mathsf{id}_1 \\ 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, b_2, \mathsf{state}_{i,2}; r_{i,2}) & \text{Otherwise} \end{cases}.$$

By our observation, $0||\tilde{y}_{\alpha,2} = 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,2}), \mathsf{state}_{i,2}; r_{i,2})$. The decryption algorithm then runs $\mathsf{HEval}(\widetilde{\mathsf{Enc\text{-}Step}}_{i,2}, \tilde{y}_{\alpha,2}, \mathsf{node}_2)$ and obtains $0||\tilde{y}_{\alpha,3} = 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{node}_{i,3}, \mathsf{state}_{i,3}; r_{i,3})$. This continues until the final

the final evaluation $\mathsf{HEval}(\widetilde{\mathsf{Enc\text{-}Step}}_{i,d}, \tilde{y}_{\alpha,d}, \mathsf{node}_d)$. As $\mathsf{id}_d = \mathsf{id}^*$ and $b_d = \mathsf{pk}^*$, the final circuit outputs $1||\mathsf{Enc}(\mathsf{pk}^*, m)$. The decryption algorithm then decrypts $\mathsf{Enc}(\mathsf{pk}^*, m)$ using PKE secret key sk^*. By the perfect correctness of PKE scheme, the decryption algorithm outputs m. ■

Remark 9.3.2. *We now describe the need of* flag *in every node of* EncTree*s present in auxilliary information. This* flag *indicates whether a node is a leaf or an internal node. We could consider an alternate construction where the node values are of the form* $a||\mathsf{id}||b \in \{0,1\}^{3\lambda}$ *(without the* flag*), and the step circuits detect any node* $a||\mathsf{id}||b$ *as a leaf by checking whether* $a = 0^\lambda$*. We notice that the construction of [48] uses this approach and does not satisfy completeness property against computationally unbounded adversaries. A computationally unbounded adversary* $\mathcal{A}$ *can break the completeness property of this alternate approach as follows.* $\mathcal{A}$ *picks an arbitrary target identity* id^* *and makes* $(\mathsf{regtgt}, \mathsf{id}^*)$ *query. The challenger samples PKE pair* $(\mathsf{pk}^*, \mathsf{sk}^*)$*, registers* $(\mathsf{id}^*, \mathsf{pk}^*)$ *and sends* pk^* *back to* $\mathcal{A}$*. At the point, the tree storing (identity, public key) pairs in* aux *consists of only a root node* rt *and a leaf node* $0^\lambda||\mathsf{id}^*||\mathsf{pk}^*$*.* $\mathcal{A}$ *then picks an arbitrary identity* $\mathsf{id} < \mathsf{id}^*$ *and computes* pk *such that* $\mathsf{Hash}(\mathsf{hk}, 0^\lambda||\mathsf{id}||\mathsf{pk}) = 0^\lambda$*.* $\mathcal{A}$ *now makes* $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$ *query. The challenger registers* $(\mathsf{id}, \mathsf{pk})$ *pair. Now, the tree consists of root node* $0^\lambda||\mathsf{id}||h$*, a left child* $0^\lambda||\mathsf{id}||\mathsf{pk}$ *and right child* $0^\lambda||\mathsf{id}^*||\mathsf{pk}^*$*. If a message is now encrypted to* id^* *and the ciphertext is decypted using update of* id^**, then the ouput is* $\perp$*.*

9.4 Security

In this section, we prove that the above scheme satisfies soundness of pre/post-Registration Verifiability and Message Hiding properties as defined in Definitions 8.3.4 to 8.3.6. We now provide a brief overview of the proofs.

Recall that soundness of pre-registration verifiability property ensures that if a PPT adversary $\mathcal{A}$ can create valid public parameters pp along with a pre-registration proof π that an identity id is not registered, then he will not be able to decrypt any ciphertext ct encrypted for id with non-negligible probability. To provide the proof's intuition, consider the scenario where a cheating accumulator/adversary creates public parameters by inserting $(\mathsf{id}, \mathsf{pk})$ at a wrong leaf location by violating property that the $\mathsf{EncTree}$ is to be sorted as per identities. Such an adversary could provide valid pre-registration proof that the identity is not registered. However, it cannot decrypt the ciphertexts encrypted for the identity. For example, the $\mathsf{EncTree}$ generated by adversary has 3 registered identities $\mathsf{id}_1 < \mathsf{id}_2 < \mathsf{id}_3$, has root value $\mathsf{rt} = h_1||\mathsf{id}_3||h_2$ with left subtree containing $\mathsf{id}_1, \mathsf{id}_3$ and right subtree containing id_2. Clearly, the paths to the leaves containing $\mathsf{id}_1, \mathsf{id}_3$ form a valid pre-registration proof. A ciphertext contains 3 garbled circuits $\{\widetilde{\mathsf{Enc\text{-}Step}_i}\}_i$ and garbling of $\mathsf{Hash}(\mathsf{rt})$. When the garbled circuit $\widetilde{\mathsf{Enc\text{-}Step}}_1$ is run with input as the root value rt, it identifies that id_2 is in left subtree (as $\mathsf{id}_2 < \mathsf{id}_3$) and outputs garbling of h_1. Now, $\widetilde{\mathsf{Enc\text{-}Step}}_2$ can only be run on the root node's left child value. The garbled values output by the garbled circuits would follow the path that is part of the pre-registration proof. As a result, the final garbled circuit outputs $\perp$,

and the adversary cannot decrypt the ciphertext. We formally prove that the scheme satisfies the property by arguing that when the adversary is forced to generate public parameters and a pre-registration proof, it cannot distinguish between a real ciphertext and a simulated ciphertext generated without using the message.

The soundness of post-registration verifiability property guarantees that if an adversary can create valid public parameters pp along with a post-registration proof π that an identity-key pair $(\mathsf{id}, \mathsf{pk})$ is registered (for an honestly generated pk such that corresponding secret key sk is not revealed to the adversary), then he will not be able to decrypt any ciphertext ct encrypted for id. The proof is similar to the proof of pre-registration verifiability, except that the simulated ciphertext is now generated using only PKE encryptions of the message with the identity's public key (the corresponding secret key is unknown to the adversary).

Message Hiding properties guarantees that if the public parameters pp are honestly generated, then a PPT adversary cannot decrypt ciphertexts of unregistered identities and cannot decrypt ciphertexts of registered identities without the knowledge of their secret keys. We argue that if any RBE scheme satisfies the soundness of pre/post-registration verifiability properties along with completeness property, it also satisfies message hiding property. If an $(\mathsf{id}, \mathsf{pk})$ pair is registered as part of pp, one could also create a valid post-registration proof as per the completeness property. Therefore, as per the soundness of post-registration verifiability, the ciphertexts meant for id cannot

be decrypted with non-negligible probability when the secret key corresponding to pk is unknown. If an id is not registered as part of pp, one could create a valid pre-registration proof as per the completeness property. Therefore, as per the soundness of pre-registration verifiability, the ciphertexts meant for id cannot be decrypted with non-negligible probability.

9.4.1 Soundness of Pre-Registration Verifiability

Theorem 9.4.1. *Assuming* HG *is a secure hash garbling scheme, the above construction satisfies the soundness of the pre-registration verifiability property (Definition 8.3.5).*

Proof. We prove the above theorem via a hybrid argument. The first hybrid H_{real} is same as the soundness game for pre-registration verifiability. In this hybrid, the challenger samples a $\mathsf{crs} = \mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$ and sends it to the adversary. The adversary then sends public parameters $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$, identity id^*, pre-registration proof π, and challenge messages m_0, m_1 to the challenger. The challenger aborts if π is invalid i.e., $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi) = 0$. Otherwise, the challenger samples a bit $b \leftarrow \{0,1\}$, encrypts message m_b and sends the ciphertext $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_i, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,1}\}_i)$ to the adversary. Concretely, given crs, public parameters pp, identity id^* and message m_b, the challenger first samples $\mathsf{state}_{i,j}, r_{i,j} \leftarrow \{0,1\}^\lambda$ for $i \in [\ell_n], j \in [d_i + 1]$. The challenger then constructs circuits $\mathsf{Enc\text{-}Step}_{i,j}$ for each $i \in [\ell_n], j \in [d_i]$ as in Figure 9.2. It then garbles the circuits $\{\mathsf{Enc\text{-}Step}_{i,j}\}_{i,j}$ along with their inputs $\{\mathsf{rt}_i\}_i$ using a hash garbling scheme

to obtain $\{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}$ and $\{\tilde{y}_{i,1}\}_i$ respectively. The adversary outputs a bit b', and wins if $b' = b$. We prove that the advantage of any PPT adversary in winning against H_{real} challenger is negligible via a sequence of hybrids.

In each subsequent hybrid, the challenger switches a garbled circuit in the ciphertext with a simulated one. Concretely, in Hybrid $H_{\kappa.\tau}$, the challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ and $\tilde{y}_{i,1}$ for each $(i,1) \preceq (\kappa,\tau)$ using a simulator $\mathsf{HG.Sim}$, whereas the challenger computes $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \succ (\kappa,\tau)$ and $\tilde{y}_{i,1}$ for each $(i,1) \succ (\kappa,\tau)$ using hash garbling scheme as earlier. Here, we say that $(i,j) \prec (k,\ell)$ if $(i < k) \vee (i = k \wedge j < \ell)$. Similarly, we say that $(i,j) \preceq (k,\ell)$ iff $(i,j) = (k,\ell) \vee (i,j) \prec (k,\ell)$. Also, $(i,j) \succ (k,\ell)$ iff $(k,\ell) \prec (i,j)$. We now describe how the challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ using a simulator.

First, we introduce some notations and make some observations. Let the proof π sent by the adversary be $\pi = (\pi_1, \cdots, \pi_{\ell_n})$, where $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$ for each $i \in [\ell_n]$. Both $\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}}$ consists of either an empty string ϵ or a sequence of values in $\{0,1\}^{3\lambda+1}$. For any $i \in [\ell_n], \mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}} = (\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$[11], where each $\mathsf{node}_{i,j,\mathsf{tag}} = \mathsf{flag}_{i,j,\mathsf{tag}} || a_{i,j,\mathsf{tag}} || \mathsf{id}_{i,j,\mathsf{tag}} || b_{i,j,\mathsf{tag}} \in \{0,1\}^{3\lambda+1}$. Let use define $\mathsf{tag}_i = \mathsf{upr}$ if $\mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$. Otherwise, let $\mathsf{tag}_i = \mathsf{lwr}$. If π is a valid pre-registration proof for id^*, then intuitively $\mathsf{path}_{i,\mathsf{tag}_i}$ resembles like a sequence of nodes obtained by performing binary search on id^*. Concretely, by Lemma 9.3.3, we know that for any index

[11] For the sake of simplicity, we assume the paths output by the adversary are of length d_i. The proof still works even if the adversary outputs paths of different lengths.

i the following holds. (1) $\mathsf{node}_{i,1,\mathsf{tag}_i} = \mathsf{rt}_i$, (2) $\mathsf{flag}_{i,j,\mathsf{tag}_i} = 1$ iff $j = d_i$, (3) $\mathsf{id}_{i,d_i,\mathsf{tag}_i} \neq \mathsf{id}^*$ and (4) for all $j < d_i$, we have

$$\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}_i}) = \begin{cases} a_{i,j,\mathsf{tag}_i} & \text{if } \mathsf{id}^* \leq \mathsf{id}_{i,j,\mathsf{tag}_i} \\ b_{i,j,\mathsf{tag}_i} & \text{if } \mathsf{id}^* > \mathsf{id}_{i,j+1,\mathsf{tag}_i} \end{cases}.$$

Consider any index $i \in [\ell_n]$. When the circuit $\mathsf{Enc\text{-}Step}_{i,1}$ (with identity id^* embedded in it) is fed with the root value $\mathsf{rt}_i = \mathsf{node}_{i,1,\mathsf{tag}_i}$ as input, the circuit outputs $\mathsf{flag}_{i,1,\mathsf{tag}_i}||\tilde{y}_{i,2} = \mathsf{flag}_{i,1,\mathsf{tag}_i}||\mathsf{HG.GarbleInp}(\mathsf{hk}, a_{i,1,\mathsf{tag}_i}, \mathsf{state}_{i,2}; r_{i,2})$ if $\mathsf{id}^* \leq \mathsf{id}_{i,1,\mathsf{tag}_i}$. Otherwise, the circuit outputs $\mathsf{flag}_{i,1,\mathsf{tag}_i}||\tilde{y}_{i,2} = \mathsf{flag}_{i,1,\mathsf{tag}_i}||\mathsf{HG.GarbleInp}(\mathsf{hk}, b_{i,1,\mathsf{tag}_i}, \mathsf{state}_{i,2}; r_{i,2})$. By the above equation, we know that $\tilde{y}_{i,2} = \mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,2}, \mathsf{state}_{i,2}; r_{i,2})$, where $h_{i,2} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,2,\mathsf{tag}_i})$. Similarly, when the circuit $\mathsf{Enc\text{-}Step}_{i,2}$ is fed with input $\mathsf{node}_{i,2,\mathsf{tag}_i}$, it outputs $\mathsf{flag}_{i,2,\mathsf{tag}_i}||\tilde{y}_{i,3} = \mathsf{flag}_{i,2,\mathsf{tag}_i}||\mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,3}, \mathsf{state}_{i,3}; r_{i,3})$, where $h_{i,3} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,3,\mathsf{tag}_i})$. On repeating this process, for each $j < d_i$, the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ outputs $\mathsf{flag}_{i,j,\mathsf{tag}_i}||\tilde{y}_{i,j+1} = \mathsf{flag}_{i,j,\mathsf{tag}_i}||\ \mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,j+1}, \mathsf{state}_{i,j+1}; r_{i,j+1})$, where $h_{i,j+1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}_i})$. The output of the final circuit $\mathsf{Enc\text{-}Step}_{i,d_i}$ is $1||\perp$ as $\mathsf{id}_{i,d_i\mathsf{tag}_i} \neq \mathsf{id}^*$.

Based on the above observations, we now describe how the challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ using a simulator. In order to simulate circuit $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}$, the challenger first samples $\mathsf{state}_{\kappa,\tau+1}, r_{\kappa,\tau+1} \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{flag}_{\kappa,\tau}||\tilde{y}_{\kappa,\tau+1}$ as follows.

$$\mathsf{flag}_{\kappa,\tau}||\tilde{y}_{\kappa,\tau+1} = \begin{cases} 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, h_{\kappa,\tau+1}, \mathsf{state}_{\kappa,\tau+1}; r_{\kappa,\tau+1}) & \text{if } \tau \neq d_\kappa \\ 1||\perp & \text{if } \tau = d_\kappa \end{cases}$$

where $h_{\kappa,\tau+1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau+1,\mathsf{tag}_\kappa})$. The challenger then runs the simulator using $\tilde{y}_{\kappa,\tau+1}$ as output and obtains

$(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}, \tilde{y}_{\kappa,\tau}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}, 1^L, \mathsf{flag}_{\kappa,\tau}||\tilde{y}_{\kappa,\tau+1})$. Here L is the length of the step circuit described in Figure 9.2. The challenger then runs the simulator again using $\tilde{y}_{\kappa,\tau}$ as output and obtains $(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau-1}, \tilde{y}_{\kappa,\tau-1}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau-1,\mathsf{tag}_\kappa}, 1^L, 0||\tilde{y}_{\kappa,\tau})$. The challenger continues the process and obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,j}, \tilde{y}_{\kappa,j}$ for each $j \leq \tau$.

For any $i < \kappa$, the challenger first sets $\tilde{y}_{i,d_i+1} = \perp$. It then runs the simulator $(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{i,j,\mathsf{tag}_i}, 1^L, \mathsf{flag}_{i,j+1}||\tilde{y}_{i,j+1})$ for each $j \leq d_i$. Here, L is the length of the step circuit described in Figure 9.2 and $\mathsf{flag}_{i,j+1} = 1$ iff $j = d_i$. The challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ and $\tilde{y}_{i,1}$ for each $(i,1) \preceq (\kappa,\tau)$ by running simulator in this way. Using the hash garbling scheme's security, we prove that every 2 adjacent hybrids are computationally indistinguishable.

In the final hybrid $H_{\ell_n.d_{\ell_n}}$, the challenger produces the entire ciphertext using a simulator without even sampling b. Clearly, the advantage of any adversary in this hybrid is 0. By the above indistinguishability argument, any PPT adversary's advantage in the original game H_{real} is negligible.

We now provide a formal description of the hybrids. Note that the numbering of hybrids depends on the size of public parameters sent by the adversary. Let us define the set S to be $\{(i,j) : i \in [\ell_n], j \in [d_i]\}$, when pp sent by the adversary is $\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$.

Hybrid H_{real}: This is the same as the original soundness game for pre-registration verifiability property.

1. The challenger first samples a hash key $\mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$, sets $\mathsf{crs} = \mathsf{hk}$ and sends it to the adversary.

2. The adversary sends an identity id^*, public parameters pp, a proof π, and a pair of messages m_0, m_1 to the challenger. The challenger aborts and the adversary looses if $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi) = 0$.

3. Let public parameters pp be $\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and proof π be $(\pi_1, \cdots, \pi_{\ell_n})$. For any $i \in [\ell_n]$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}}$ be sequence of values $(\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$.[12]

4. Let $S = \{(i,j) : i \in [\ell_n], j \in [d_i]\}$ and $S' = \{(i,j) : i \in [\ell_n], j \in [d_i + 1]\}$. The challenger samples a bit $b \leftarrow \{0,1\}$.

5. It then samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda, r_{i,j} \leftarrow \{0,1\}^\lambda$ for each $(i,j) \in S'$.

6. For each $(i,j) \in S$, the challenger defines the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as in Figure 9.2 with hash key hk, state $\mathsf{state}_{i,j+1}$, randomness $r_{i,j+1}$, identity id^*, message m_b hardwired in it. It then garbles the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ i.e., $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{i,j}, \mathsf{state}_{i,j})$. For each $i \in [\ell_n]$, the challenger computes $\tilde{y}_{i,1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,1}, \mathsf{state}_{i,1}; r_{i,1})$, where $h_{i,1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{rt}_i)$.

7. Finally, the challenger sends ciphertext $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{(i,j)\in S}, \{\tilde{y}_{i,1}\}_{i\in[\ell_n]})$ to the adversary.

[12] To simplify the notations, we assume that the paths $\mathsf{path}_{i,\mathsf{lwr}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ output by the adversary have length d_i. With appropriate changes, the proof works even if length of the paths is smaller than d_i.

8. The adversary outputs a bit b'. The adversary wins if $b' = b$.

Hybrid $H_{\kappa.\tau}$ ($(\kappa, \tau) \in S$): In this hybrid, for every $(i, j) \preceq (\kappa, \tau)$, the challenger computes $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ using a simulator. The changes from the previous hybrid are highlighted in red.

5. It then samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda, r_{i,j} \leftarrow \{0,1\}^\lambda$ for each $(i, j) \in S'$ s.t. $(i, j) \succ (\kappa, \tau)$.

6. For each $(i, j) \in S$ s.t. $(i, j) \succ (\kappa, \tau)$, the challenger defines the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as in Figure 9.2 with hash key hk, state $\mathsf{state}_{i,j+1}$, randomness $r_{i,j+1}$, identity id^*, message m_b hardwired in it. It then garbles the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ i.e., $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{i,j}, \mathsf{state}_{i,j})$.

   For each i s.t $(i, 1) \succ (\kappa, \tau)$, the challenger computes $\tilde{y}_{i,1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,1}, \mathsf{state}_{i,1}; r_{i,1})$, where $h_{i,1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{rt}_i)$. For any i, let use define $\mathsf{tag}_i = \mathsf{upr}$ if $\mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$. Otherwise, let $\mathsf{tag}_i = \mathsf{lwr}$. For each i s.t. $(i, d_i) \preceq (\kappa, \tau)$, the challenger sets $\tilde{y}_{i,d_i+1} = \perp$. If $\tau \neq d_\kappa$, the challenger computes $x = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau+1,\mathsf{tag}_i})$ and sets $\tilde{y}_{\kappa,\tau+1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, x, \mathsf{state}_{\kappa,\tau+1}; r_{\kappa,\tau+1})$. For each $(i, j) \in S$ s.t $(i, j) \preceq (\kappa, \tau)$, the challenger simulates the garbling of $\mathsf{Enc\text{-}Step}_{i,j}$ as $(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{i,j,\mathsf{tag}_i}, 1^L, \mathsf{flag}_{i,j+1} || \tilde{y}_{i,j+1})$. Here L is the length of circuit $\mathsf{Enc\text{-}Step}_{i,j}$ defined in Figure 9.2 and $\mathsf{flag}_{i,j+1} = 1$ iff $j = d_i$. [13]

[13] Note that simulating $\tilde{y}_{i,j}$ requires $\tilde{y}_{i,j+1}$. As a result, for each i, the challenger runs the

For the sake of simplicity, we define Hybrid $H_{1.0}$ same as Hybrid H_{real}. Note that hybrid H_{real} ($H_{1.0}$) is the same as the soundness game for pre-registration verifiability. In this hybrid, the challenger generates the ciphertext by garbing circuits $\mathsf{Enc\text{-}Step}_{i,j}$ and their inputs rt_i using a hash garbling scheme. In the final hybrid $H_{\ell_n.d_{\ell_n}}$, the challenger generates the ciphertext using simulator given only public parameters and length of the step circuits. The ciphertext in this final hybrid is independent of b. Consequently, the advantage of any adversary in Hybrid $H_{\ell_n.d_{\ell_n}}$ is 0. We prove that hybrids H_{real} and $H_{\ell_n.d_{\ell_n}}$ are computationally indistinguishable via a sequence of intermediate hybrids $H_{\kappa.\tau}$. For any PPT adversary $\mathcal{A}$, let the advantage of $\mathcal{A}$ in Hybrid H_s be $\mathsf{Adv}_s^{\mathcal{A}} = \Pr[b' = b] - 1/2$. For any $\kappa \in [\ell_n], \tau \in [d_\kappa]$, let $(\tilde{\kappa}, \tilde{\tau})$ be its preceding tuple i.e., $(\tilde{\kappa}, \tilde{\tau}) = \begin{cases} (\kappa, \tau - 1) & \text{if } \tau > 1 \\ (\kappa - 1, d_{\kappa-1}) & \text{if } \kappa > 1 \wedge \tau = 1 \\ (1, 0) & \text{if } (\kappa, \tau) = (1, 1) \end{cases}$.

Lemma 9.4.1. *Assuming* HG *is a secure hash garbling scheme, for every admissible*[14] *PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$ *and any* $(\kappa, \tau) \in S$*, we have* $|\mathsf{Adv}_{\kappa.\tau}^{\mathcal{A}} - \mathsf{Adv}_{\tilde{\kappa}.\tilde{\tau}}^{\mathcal{A}}| \leq negl(\lambda)$.

Proof. Suppose there exists an admissible PPT adversary $\mathcal{A}$, indices $(\kappa, \tau) \in S$ such that $|\mathsf{Adv}_{\kappa.\tau}^{\mathcal{A}} - \mathsf{Adv}_{\tilde{\kappa}.\tilde{\tau}}^{\mathcal{A}}|$ is a non-negligible value ϵ. We construct a non-uniform reduction algorithm $\mathcal{B}$ that breaks the security of hash garbling scheme.

simulator in the decreasing order of j.

[14] A PPT adversary in the soundness game is said to be admissible if it always produces valid proof π.

The hash garbling game challenger $\mathcal{C}$ first sends a hash key hk to the reduction algorithm $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{crs} = \mathsf{hk}$ and sends it to the adversary $\mathcal{A}$. The adversary sends an identity id^*, public parameters pp, a proof π and a pair of messages m_0, m_1 to $\mathcal{B}$. If $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi) = 0$, $\mathcal{B}$ aborts and outputs a random bit in hash garbling game. Otherwise let $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$, $\pi = (\pi_1, \cdots, \pi_{\ell_n})$. For any $i \in [\ell_n]$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}}$ be sequence of values $(\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$[12]. Let $S = \{(i,j) : i \in [\ell_n], j \in [d_i]\}$ and $S' = \{(i,j) : i \in [\ell_n], j \in [d_i + 1]\}$. For any i, let use define $\mathsf{tag}_i = \mathsf{upr}$ if $\mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$. Otherwise, let $\mathsf{tag}_i = \mathsf{lwr}$. $\mathcal{B}$ then samples a bit $b \leftarrow \{0,1\}$. For each $(i,j) \in S'$ s.t. $(i,j) \succ (\kappa,\tau)$, $\mathcal{B}$ samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda, r_{i,j} \leftarrow \{0,1\}^\lambda$. For each $(i,j) \in S$ s.t. $(i,j) \succeq (\kappa,\tau)$, $\mathcal{B}$ creates the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as in Figure 9.2 with identity id^*, message m_b hardwired in it. $\mathcal{B}$ then sends $\mathsf{Enc\text{-}Step}_{\kappa,\tau}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}$ to the challenger $\mathcal{C}$. $\mathcal{C}$ samples a bit $\gamma \leftarrow \{0,1\}$. If $\gamma = 0$, $\mathcal{C}$ samples $\mathsf{state} \leftarrow \{0,1\}^\lambda$ and computes $h = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa})$, $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{\kappa,\tau}, \mathsf{state})$, $\tilde{y}_{\kappa,\tau} \leftarrow \mathsf{HG.GarbleInp}(\mathsf{hk}, h, \mathsf{state})$. Otherwise $\mathcal{C}$ computes $x = \mathsf{Enc\text{-}Step}_{\kappa,\tau}(\mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa})$ and runs simulator $(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}, \tilde{y}_{\kappa,\tau}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}, 1^{|\mathsf{Enc\text{-}Step}_{\kappa,\tau}|}, x)$. $\mathcal{C}$ sends $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}$, $\tilde{y}_{\kappa,\tau}$ to $\mathcal{B}$. For each $(i,j) \in S$ s.t. $(i,j) \succ (\kappa,\tau)$, $\mathcal{B}$ garbles the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ to obtain $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{i,j}, \mathsf{state}_{i,j})$. For each i s.t $(i,1) \succ (\kappa,\tau)$, $\mathcal{B}$ garbles the input rt_i and obtains $\tilde{y}_{i,1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{rt}_i, \mathsf{state}_{i,1}; r_{i,1})$. For each $(i,j) \in S$ s.t. $(i,j) \prec (\kappa,\tau)$, $\mathcal{B}$ runs the simulator and computes $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}$ as in Hybrid $H_{\kappa.\tau}$. Finally, $\mathcal{B}$ sends the cipher-

text $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_i, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,1}\}_i)$ to $\mathcal{A}$. $\mathcal{A}$ outputs a bit b'. If $b = b'$, $\mathcal{B}$ outputs 1 in the hash garbling game. Otherwise $\mathcal{B}$ outputs 0 in the hash garbling game.

We now analyze the advantage of $\mathcal{B}$ in hash garbling game. As the adversary $\mathcal{A}$ is admissible, $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi) = 1$ and $\mathcal{B}$ does not abort. By Lemma 9.3.3, when $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \pi) = 1$, the path $\mathsf{path}_{\kappa,\mathsf{tag}_\kappa}$ resembles a sequence of nodes obtain by performing binary search on id^* in a tree with root rt_κ. Consequently, when the challenger runs step circuit $\mathsf{Enc\text{-}Step}_{\kappa,\tau}$ on input $\mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}$, the result is $0||\mathsf{node}_{\kappa,\tau+1,\mathsf{tag}_\kappa}$ if $\tau < d_\kappa$, and $1||\perp$ if $\tau = d_\kappa$. As a result, if $\gamma = 0$, then $\mathcal{B}$ emulates Hybrid $H_{\tilde{\kappa}.\tilde{\tau}}$ challenger to $\mathcal{A}$, and if $\gamma = 1$, then $\mathcal{B}$ emulates Hybrid $H_{\kappa.\tau}$ challenger to $\mathcal{A}$. By our assumption, $|\mathsf{Adv}^{\mathcal{A}}_{\kappa.\tau} - \mathsf{Adv}^{\mathcal{A}}_{\tilde{\kappa}.\tilde{\tau}}|$ is non-negligible. Therefore, $\mathcal{B}$ has a non-negligible advantage in predicting γ.

■

By the above lemma and triangle inequality, the advantage of any PPT adversary in Hybrid H_{real} is negligible. Therefore, our construction satisfies soundness for pre-registration verifiability property. ■

9.4.2 Soundness of Post-Registration Verifiability

Theorem 9.4.2. *Assuming* HG *is a secure hash garbling scheme and* PKE *is a secure public-key encryption scheme, the above construction satisfies the soundness of post-registration verifiability property (Definition 8.3.5).*

We prove the above theorem via a hybrid argument. In the first hybrid H_b ($b \in \{0,1\}$), the challenger samples a crs, a PKE pair $(\mathsf{pk}, \mathsf{sk})$ and sends a $(\mathsf{crs}, \mathsf{pk})$ to the adversary. The adversary then sends public parameters $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$, identity id^*, post-registration proof π, and challenge messages m_0, m_1 to the challenger. The challenger aborts if π is invalid i.e., $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}, \pi) = 0$. Otherwise, the challenger encrypts message m_b and sends the ciphertext $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_i, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}, \{\tilde{y}_{i,1}\}_i)$ to the adversary. Concretely, given crs, public parameters pp, identity id^* and message m_b, the challenger first samples $\mathsf{state}_{i,j}, r_{i,j} \leftarrow \{0,1\}^\lambda$ for each $i \in [\ell_n], j \in [d_i + 1]$. It then constructs circuits $\mathsf{Enc\text{-}Step}_{i,j}$ for each $i \in [\ell_n], j \in [d_i]$ as in Figure 9.2. The challenger then garbles the circuits $\{\mathsf{Enc\text{-}Step}_{i,j}\}_{i,j}$ along with their inputs $\{\mathsf{rt}_i\}_i$ using a hash garbling scheme to obtain $\{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i,j}$ and $\{\tilde{y}_{i,1}\}_i$ respectively. Finally, the adversary outputs bit b'. We prove that for any PPT adversary, $\Pr[b' = 1]$ in Hybrids H_0 and H_1 are negligibly close via a sequence of hybrids.

In each subsequent hybrid, the challenger switches a garbled circuit in the ciphertext with a simulated one. Concretely, in Hybrid $H_{\kappa.\tau}$, the challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ and $\tilde{y}_{i,1}$ for each $(i,1) \preceq (\kappa,\tau)$ using a simulator. The challenger computes $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \succ (\kappa,\tau)$ and $\tilde{y}_{i,1}$ for each $(i,1) \succ (\kappa,\tau)$ using hash garbling scheme as earlier. Here, we say that $(i,j) \prec (k,\ell)$ if $(i < k) \vee (i = k \wedge j < \ell)$. Similarly, we say that $(i,j) \preceq (k,\ell)$ iff $(i,j) = (k,\ell) \vee (i,j) \prec (k,\ell)$. Also, $(i,j) \succ (k,\ell)$ iff $(k,\ell) \prec (i,j)$. We now describe how the challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ using a

simulator.

First, we introduce some notations and make some observations. Let the proof π output by the adversary be $\pi = (\pi_1, \cdots, \pi_{\ell_n}, \alpha)$. For any $i < \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \geq \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \in [\ell_n], \mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}} = (\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$[15], where each $\mathsf{node}_{i,j,\mathsf{tag}} = \mathsf{flag}||a_{i,j,\mathsf{tag}}||\mathsf{id}_{i,j,\mathsf{tag}}||b_{i,j,\mathsf{tag}} \in \{0,1\}^{3\lambda+1}$. For any $i \in [\ell_n]$, let us define

$$\mathsf{tag}_i = \begin{cases} \mathsf{upr} & \text{if } i < \alpha \wedge \mathsf{path}_{i,\mathsf{upr}} \neq \epsilon \\ \mathsf{lwr} & \text{if } i < \alpha \wedge \mathsf{path}_{i,\mathsf{upr}} = \epsilon \\ \mathsf{mid} & \text{Otherwise} \end{cases}.$$

Suppose π is a valid post-registration proof for id^*. We know that for any i, $\mathsf{path}_{i,\mathsf{tag}_i}$ looks like a sequence of nodes obtained by performing a binary search on id^*. Concretely, by Lemmas 9.3.3 and 9.3.4, we know that for any index i, the following holds. (1) $\mathsf{node}_{i,1,\mathsf{tag}_i} = \mathsf{rt}_i$, (2) $\mathsf{flag}_{i,j,\mathsf{tag}_i} = 1$ iff $j = d_i$, (3) $\mathsf{id}_{i,d_i,\mathsf{tag}_i} \neq \mathsf{id}^*$ if $i < \alpha$, (4) $\mathsf{id}_{i,d_i,\mathsf{tag}_i} = \mathsf{id}^*$, $b_{i,d_i,\mathsf{tag}_i} = \mathsf{pk}^*$ if $i \geq \alpha$, and (5) for all $j < d_i$,

$$\mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}_i}) = \begin{cases} a_{i,j,\mathsf{tag}_i} & \text{if } \mathsf{id}^* \leq \mathsf{id}_{i,j,\mathsf{tag}_i} \\ b_{i,j,\mathsf{tag}_i} & \text{if } \mathsf{id}^* > \mathsf{id}_{i,j,\mathsf{tag}_i} \end{cases}.$$

Consider any index $i \in [\ell_n]$. When the circuit $\mathsf{Enc\text{-}Step}_{i,1}$ (with identity id^* embedded in it) is fed with the root value $\mathsf{rt}_i = \mathsf{node}_{i,1,\mathsf{tag}_i}$ as input, the

[15]To simplify the notations, we assume that the paths $\mathsf{path}_{i,\mathsf{lwr}}$, $\mathsf{path}_{i,\mathsf{mid}}$ and $\mathsf{path}_{i,\mathsf{upr}}$ output by the adversary have length d_i. With appropriate changes, the proof works even if length of the paths is smaller than d_i.

circuit outputs

$$\mathsf{flag}_{i,1,\mathsf{tag}_i}||\tilde{y}_{i,2} = \begin{cases} \mathsf{flag}_{i,1,\mathsf{tag}_i}||\mathsf{HG.GarbleInp}(\mathsf{hk}, a_{i,1,\mathsf{tag}_i}, \mathsf{state}_{i,2}; r_{i,2}) & \text{if } \mathsf{id}^* \leq \mathsf{id}_{i,1,\mathsf{tag}_i} \\ \mathsf{flag}_{i,1,\mathsf{tag}_i}||\mathsf{HG.GarbleInp}(\mathsf{hk}, b_{i,1,\mathsf{tag}_i}, \mathsf{state}_{i,2}; r_{i,2}) & \text{Otherwise} \end{cases}$$
$$= \mathsf{flag}_{i,1,\mathsf{tag}_i}||| \mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,2}, \mathsf{state}_{i,2}; r_{i,2})$$

where $h_{i,2} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,2,\mathsf{tag}_i})$. Similarly, when the circuit $\mathsf{Enc\text{-}Step}_{i,2}$ is fed with input $\mathsf{node}_{i,2,\mathsf{tag}_i}$, it outputs $\mathsf{flag}_{i,2,\mathsf{tag}_i}||\tilde{y}_{i,3} = \mathsf{flag}_{i,2,\mathsf{tag}_i}||\mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,3}, \mathsf{state}_{i,3}; r_{i,3})$, where $h_{i,3} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,3,\mathsf{tag}_i})$. On repeating this process, for each $j < d_i$, the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ outputs $\mathsf{flag}_{i,j,\mathsf{tag}_i}||\tilde{y}_{i,j+1} = \mathsf{flag}_{i,j,\mathsf{tag}_i}||$ $\mathsf{HG.GarbleInp}(\mathsf{hk}, h_{i,j+1}, \mathsf{state}_{i,j+1}; r_{i,j+1})$, where $h_{i,j+1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{i,j+1,\mathsf{tag}_i})$. The output of the final circuit $\mathsf{Enc\text{-}Step}_{i,d_i}$ is $1||\perp$ if $i < \alpha$, and $1||\mathsf{Enc}(\mathsf{pk}, m)$ if $i \geq \alpha$.

Based on the above observations, we now describe how the challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ using a simulator. In order to simulate circuit $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}$, the challenger first samples $\mathsf{state}_{\kappa,\tau+1}, r_{\kappa,\tau+1} \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{flag}_{\kappa,\tau}||\tilde{y}_{\kappa,\tau+1}$ as follows.

$$\mathsf{flag}_{\kappa,\tau}||\tilde{y}_{\kappa,\tau+1} = \begin{cases} 0||\mathsf{HG.GarbleInp}(\mathsf{hk}, h_{\kappa,\tau+1}, \mathsf{state}_{\kappa,\tau+1}; r_{\kappa,\tau+1}) & \text{if } \tau \neq d_\kappa \\ 1||\perp & \text{if } \tau = d_\kappa \wedge \kappa < \alpha \\ 1||\mathsf{Enc}(\mathsf{pk}, m_b) & \text{if } \tau = d_\kappa \wedge \kappa \geq \alpha \end{cases}$$

where $h_{\kappa,\tau+1} = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau+1,\mathsf{tag}_i})$. The challenger then runs the simulator using $\tilde{y}_{\kappa,\tau+1}$ as output and obtains $(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}, \tilde{y}_{\kappa,\tau}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_i}, 1^L, \mathsf{flag}_{\kappa,\tau}||\tilde{y}_{\kappa,\tau+1})$. Here L is the length of the step circuit described in Figure 9.2. The challenger then runs the simulator again using $\tilde{y}_{\kappa,\tau}$ as output and obtains $(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau-1}, \tilde{y}_{\kappa,\tau-1}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau-1,\mathsf{tag}_i}, 1^L, 0||\tilde{y}_{\kappa,\tau})$.

Similarly, for each $j < \tau$, the challenger samples $(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,j}, \tilde{y}_{\kappa,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,j,\mathsf{tag}_i}, 1^L, 0||\tilde{y}_{\kappa,j+1})$.

For any $i < \kappa$, the challenger first sets $\tilde{y}_{i,d_i+1} = \begin{cases} \bot & \text{if } i < \alpha \\ \mathsf{Enc}(\mathsf{pk}, m_b) & \text{Otherwise} \end{cases}$. It then runs the simulator $(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{i,j,\mathsf{tag}_i}, 1^L, \mathsf{flag}_{i,j}||\tilde{y}_{i,j+1})$ for each $j \leq d_\kappa$. Here, L is the length of the step circuit described in Figure 9.2 and $\mathsf{flag}_{i,j+1} = 1$ iff $j = d_i$. The challenger obtains $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}$ for each $(i,j) \preceq (\kappa,\tau)$ and $\tilde{y}_{i,1}$ for each $(i,1) \preceq (\kappa,\tau)$ by running simulator in this way. Using the hash garbling scheme's security, we prove that every 2 adjacent hybrids are computationally indistinguishable.

In the final hybrid $H_{b.\ell_n.d_{\ell_n}}$, the challenger produces the entire ciphertext using a simulator given PKE encryptions of message m_b. By semantic security of the underlying PKE system, we know that PKE encryptions of m_0 are computationally indistinguishable from PKE encryptions of m_1. Using this, we prove that hybrids $H_{0.\ell_n.d_{\ell_n}}$ and $H_{1.\ell_n.d_{\ell_n}}$ are computationally indistinguishable. By combining the above indistinguishability arguments, we prove that Hybrids H_0 and H_1 are computationally indistinguishable.

We now provide a formal description of the hybrids. Note that the numbering of hybrids depends on the size of public parameters sent by the adversary. Let us define the set S to be $\{(i,j) : i \in [\ell_n], j \in [d_i]\}$, when pp sent by the adversary is $\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$.

Proof. We prove the above scheme using a sequence of following hybrids.

Hybrid H_b ($b \in \{0,1\}$): This hybrid is same as the original soundness game in which the challenger always encrypts message m_b.

1. The challenger first samples a hash key $\mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$, a PKE key pair $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, sets $\mathsf{crs} = \mathsf{hk}$ and sends $(\mathsf{crs}, \mathsf{pk}^*)$ to the adversary.

2. The adversary sends an identity id^*, public parameters pp, a proof π and a pair of messages m_0, m_1 to the challenger. The challenger aborts and the adversary looses if $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 0$.

3. Let public parameters $\mathsf{pp} = \{(\mathsf{rt}_i, d_i)\}_{i \in [\ell_n]}$ and proof $\pi = (\pi_1, \cdots, \pi_{\ell_n}, \alpha)$. Let $S = \{(i,j) : i \in [\ell_n], j \in [d_i]\}$ and $S' = \{(i,j) : i \in [\ell_n], j \in [d_i + 1]\}$.

4. For each $i < \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{upr}})$. Similarly, for each $i \geq \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}}$ be a sequence of values $(\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$.[15]. For any $i \in [\ell_n]$, let $\mathsf{tag}_i = \begin{cases} \mathsf{upr} & \text{if } i < \alpha \\ \mathsf{mid} & \text{Otherwise} \end{cases}$.

5. The challenger samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda, r_{i,j} \leftarrow \{0,1\}^\lambda$ for each $(i,j) \in S'$.

6. For each $(i,j) \in S$, the challenger defines the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as in Figure 9.2 with hash key hk, state $\mathsf{state}_{i,j+1}$, randomness $r_{i,j+1}$, identity id^*, message m_b hardwired in it. It then garbles the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ i.e., $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{i,j}, \mathsf{state}_{i,j})$.

For each $i \in [\ell_n]$, the challenger computes $\tilde{y}_{i,1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{rt}_i), \mathsf{state}_{i,1}; r_{i,1})$.

7. Finally, the challenger sends ciphertext $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{(i,j)\in S}, \{\tilde{y}_{i,1}\}_{i\in[\ell_n]})$ to $\mathcal{A}$. The adversary outputs a bit b'.

Hybrid $H_{b.\kappa.\tau}$ $(b \in \{0,1\}, (\kappa,\tau) \in S)$: In this hybrid, for every $(i,j) \preceq (\kappa,\tau)$, the challenger computes $\mathsf{Enc\text{-}Step}i, j$ using a simulator. The differences from the previous hybrid are highlighted in red color.

5. The challenger samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda, r_{i,j} \leftarrow \{0,1\}^\lambda$ for each $(i,j) \in S'$ s.t. $(i,j) \succ (\kappa,\tau)$.

6. For each $(i,j) \in S$ s.t. $(i,j) \succ (\kappa,\tau)$, the challenger defines the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as in Figure 9.2 with hash key hk, state $\mathsf{state}_{i,j+1}$, randomness $r_{i,j+1}$, identity id^*, message m_b hardwired in it. It then garbles the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ i.e., $\widetilde{\mathsf{Enc\text{-}Step}}_{i,j} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{i,j}, \mathsf{state}_{i,j})$.

   For each $i \in [\ell_n]$ s.t $(i,1) \succ (\kappa,\tau)$, the challenger computes $\tilde{y}_{i,1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{rt}_i), \mathsf{state}_{i,1}; r_{i,1})$.

   For each i s.t. $(i, d_i) \preceq (\kappa,\tau)$,

   - The challenger sets $\tilde{y}_{i,d_i+1} = 1 || \perp$ if $i < \alpha$.
   - The challenger samples ciphertext $\mathsf{ct}_i^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}^*, m_b)$ and sets $\tilde{y}_{i,d_i+1} = 1 || \mathsf{ct}_i^*$ if $i \geq \alpha$.

For any i, Let us define $\mathsf{tag}_i = \mathsf{upr}$ if $i < \alpha \wedge \mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$. $\mathsf{tag}_i = \mathsf{lwr}$ if $i < \alpha \wedge \mathsf{path}_{i,\mathsf{upr}} = \epsilon$, and $\mathsf{tag}_i = \mathsf{mid}$ if $i \geq \alpha$.

If $\tau \neq d_\kappa$, the challenger computes $x = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau+1,\mathsf{tag}_\kappa})$ and sets $\tilde{y}_{\kappa,\tau+1} = \mathsf{HG.GarbleInp}(\mathsf{hk}, x, \mathsf{state}_{\kappa,\tau+1}; r_{\kappa,\tau+1})$.

For each $(i,j) \in S$ s.t $(i,j) \preceq (\kappa,\tau)$, the challenger simulates the garbling of $\mathsf{Enc\text{-}Step}_{i,j}$ as $(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{i,j,\mathsf{tag}_i}, 1^L, \mathsf{flag}_{i,j+1}||\tilde{y}_{i,j+1})$[16], where L is the length of circuit $\mathsf{Enc\text{-}Step}_{i,j}$ defined in Figure 9.2 and $\mathsf{flag}_{i,j+1} = 1$ iff $j = d_i$.

For the sake of simplicity, we hereby define Hybrid $H_{b.1.0}$ same as Hybrid H_b for any bit b. We now prove that Hybrid H_0 is computationally indistinguishable from Hybrid H_1 via a sequence of intermediate hybrids. For any PPT adversary $\mathcal{A}$ and any Hybrid H_s, let the probability that $\mathcal{A}$ outputs 1 in Hybrid H_s be $p_s^{\mathcal{A}}$. For any $\kappa \in [\ell_n], \tau \in [d_\kappa]$, let $(\tilde{\kappa}, \tilde{\tau})$ be its preceding tuple i.e.,

$$(\tilde{\kappa}, \tilde{\tau}) = \begin{cases} (\kappa, \tau - 1) & \text{if } \tau > 1 \\ (\kappa - 1, d_{\kappa-1}) & \text{if } \kappa > 1 \wedge \tau = 1\,. \\ (1,0) & \text{if } (\kappa,\tau) = (1,1) \end{cases}$$

Lemma 9.4.2. *Assuming* HG *is a secure hash garbling scheme, for every admissible[14] PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for any $\lambda \in \mathbb{N}$, bit $b \in \{0,1\}$, any $(\kappa,\tau) \in S$, we have $|p^{\mathcal{A}}_{b.\kappa.\tau} - p^{\mathcal{A}}_{b.\tilde{\kappa}.\tilde{\tau}}| \leq negl(\lambda)$.*

Proof. Suppose there exists an admissible PPT adversary $\mathcal{A}$ and indices $(\kappa,\tau) \in$

[16]Note that simulating $\tilde{y}_{i,j}$ requires $\tilde{y}_{i,j+1}$. As a result, for each i, the challenger runs the simulator in the decreasing order of j.

S s.t. $|p^{\mathcal{A}}_{b.\kappa.\tau} - p^{\mathcal{A}}_{b.\tilde{\kappa}.\tilde{\tau}}|$ is a non-negligible value ϵ. We construct a non-uniform reduction algorithm $\mathcal{B}$ which knows the indices (κ, τ) and breaks the security of hash garbling scheme.

The hash garbling game challenger $\mathcal{C}$ first sends a hash key hk to the reduction algorithm $\mathcal{B}$. $\mathcal{B}$ samples a PKE key pair $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, sets $\mathsf{crs} = \mathsf{hk}$ and sends $(\mathsf{crs}, \mathsf{pk}^*)$ to the adversary $\mathcal{A}$. The adversary sends an identity id^*, public parameters pp, a proof π and a pair of messages m_0, m_1 to the challenger. If $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 0$, $\mathcal{B}$ aborts and outputs a random bit in hash garbling game. Let public parameters pp be $\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and proof π be $(\pi_1, \cdots, \pi_{\ell_n}, \alpha)$. For each $i < \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}})$. Similarly, for each $i \geq \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}}$ be a sequence of values $(\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$[15]. Let us define set $S = \{(i,j) : i \in [\ell_n], j \in [d_i]\}$ and set $S' = \{(i,j) : i \in [\ell_n], j \in [d_i + 1]\}$. For any i, Let us define $\mathsf{tag}_i = \mathsf{upr}$ if $i < \alpha \wedge \mathsf{path}_{i,\mathsf{upr}} \neq \epsilon$. $\mathsf{tag}_i = \mathsf{lwr}$ if $i < \alpha \wedge \mathsf{path}_{i,\mathsf{upr}} = \epsilon$, and $\mathsf{tag}_i = \mathsf{mid}$ if $i \geq \alpha$. For each $(i,j) \in S'$ s.t. $(i,j) \succ (\kappa, \tau)$, $\mathcal{B}$ samples $\mathsf{state}_{i,j} \leftarrow \{0,1\}^\lambda, r_{i,j} \leftarrow \{0,1\}^\lambda$. For each $(i,j) \in S$ s.t. $(i,j) \succeq (\kappa, \tau)$, $\mathcal{B}$ creates the circuit $\mathsf{Enc\text{-}Step}_{i,j}$ as in Figure 9.2 with message m_b and identity id^* hardwired in it. $\mathcal{B}$ then sends circuit $\mathsf{Enc\text{-}Step}_{\kappa,\tau}$ and input $\mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}$ to the hash garbling game challenger $\mathcal{C}$. $\mathcal{C}$ samples a bit $\gamma \leftarrow \{0,1\}$. If $\gamma = 0$, $\mathcal{C}$ samples $\mathsf{state} \leftarrow \{0,1\}^\lambda$ and computes $h = \mathsf{HG.Hash}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa})$, $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau} \leftarrow \mathsf{HG.GarbleCkt}(\mathsf{hk}, \mathsf{Enc\text{-}Step}_{\kappa,\tau}, \mathsf{state})$ and $\tilde{y}_{\kappa,\tau} \leftarrow \mathsf{HG.GarbleInp}(\mathsf{hk}, h, \mathsf{state})$. Otherwise $\mathcal{C}$ computes $x = \mathsf{Enc\text{-}Step}_{\kappa,\tau}(\mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa})$ and runs simulator $(\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau},$

$\tilde{y}_{\kappa,\tau}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}, 1^{|\mathsf{Enc\text{-}Step}_{\kappa,\tau}|}, x)$. $\mathcal{C}$ sends $\widetilde{\mathsf{Enc\text{-}Step}}_{\kappa,\tau}, \tilde{y}_{\kappa,\tau}$ to $\mathcal{B}$. For each $i < \alpha$ s.t. $(i, d_i) \prec (\kappa, \tau)$, the challenger sets $\tilde{y}_{i,d_i+1} = 1||\perp$. For each $i \geq \alpha$ s.t. $(i, d_i) \prec (\kappa, \tau)$, $\mathcal{B}$ samples a ciphertext $\mathsf{ct}_i^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}^*, m_b)$ and sets $\tilde{y}_{i,j+1} = 1||\mathsf{ct}_i^*$. For each $(i, j) \in S$ s.t $(i, j) \prec (\kappa, \tau)$, $\mathcal{B}$ simulates the garbling of $\mathsf{Enc\text{-}Step}_{i,j}$ as $(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{i,j}, 1^L, \mathsf{flag}_{i,j+1}||\tilde{y}_{i,j+1})$[13], where L is the length of circuit $\mathsf{Enc\text{-}Step}_{i,j}$ defined in Figure 9.2 and $\mathsf{flag}_{i,j+1} = 1$ iff $j = d_i$. Finally, $\mathcal{B}$ sends ciphertext $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{(i,j)\in S}, \{\tilde{y}_{i,1}\}_{i\in[\ell_n]})$ to $\mathcal{A}$. The adversary outputs a bit b'. $\mathcal{B}$ outputs b' as its guess in hash garbling game.

We now analyze the advantage of $\mathcal{B}$ in hash garbling game. As the adversary $\mathcal{A}$ is admissible, $\mathsf{PostVerify}\ (\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 1$ and $\mathcal{B}$ does not abort. By Lemmas 9.3.3 and 9.3.4, when $\mathsf{PostVerify}(\mathsf{crs},\ \mathsf{pp}, \mathsf{id}^*, \pi) = 1$, the path $\mathsf{path}_{\kappa,\mathsf{tag}_\kappa}$ resembles a sequence of nodes obtain by performing binary search on id^* in a tree with root rt_κ. Consequently, when the challenger runs step circuit $\mathsf{Enc\text{-}Step}_{\kappa,\tau}$ on input $\mathsf{node}_{\kappa,\tau,\mathsf{tag}_\kappa}$, the result is $0||\mathsf{node}_{\kappa,\tau+1,\mathsf{tag}_\kappa}$ if $\tau < d_\kappa$, and $1||\perp$ if $\tau = d_\kappa \wedge \kappa < \alpha$, and $1||\mathsf{Enc}(\mathsf{pk}^*, m_b)$ if $\tau = d_\kappa \wedge \kappa \geq \alpha$. As a resut, if $\gamma = 0$, then $\mathcal{B}$ emulates Hybrid $H_{b.\tilde{\kappa}.\tilde{\tau}}$ challenger to $\mathcal{A}$ and if $\gamma = 1$, then $\mathcal{B}$ emulates Hybrid $H_{b.\kappa.\tau}$ challenger to $\mathcal{A}$. By our assumption, $|p^{\mathcal{A}}_{\kappa.\tau} - p^{\mathcal{A}}_{\tilde{\kappa}.\tilde{\tau}}|$ is non-negligible. Therefore, the advantage of $\mathcal{B}$ in hash garbling game given by $|\Pr[b' = 1|\gamma = 0] - \Pr[b' = 1|\gamma = 1]|$ is non-negligible. ■

Lemma 9.4.3. *Assuming* PKE *is a secure public key encryption scheme, for every admissible*[14] *PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$*, we have* $|p^{\mathcal{A}}_{0.\ell_n.d_{\ell_n}} - p^{\mathcal{A}}_{1.\ell_n.d_{\ell_n}}| \leq negl(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|p^{\mathcal{A}}_{0.\ell_n.d_{\ell_n}} - p^{\mathcal{A}}_{1.\ell_n.d_{\ell_n}}|$ is non-negligible value ϵ. We construct a reduction algorithm $\mathcal{B}$ that breaks the semantic security of the PKE scheme.[17]

The challenger first sends a public key pk^* to the reduction algorithm $\mathcal{B}$. $\mathcal{B}$ samples a hash key $\mathsf{hk} \leftarrow \mathsf{HG.Setup}(1^\lambda, 1^{3\lambda+1})$, sets $\mathsf{crs} = \mathsf{hk}$ and sends $(\mathsf{crs}, \mathsf{pk}^*)$ to the adversary. The adversary then sends an identity id^*, public parameters pp, a proof π and a pair of messages m_0, m_1 to $\mathcal{B}$. If $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi) = 0$, $\mathcal{B}$ aborts and outputs a random bit in PKE game. Let public parameters pp be $\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}$ and proof π be $(\pi_1, \cdots, \pi_{\ell_n}, \alpha)$. For each $i < \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}})$. Similarly, for each $i \geq \alpha$, let $\pi_i = (\mathsf{path}_{i,\mathsf{lwr}}, \mathsf{path}_{i,\mathsf{mid}}, \mathsf{path}_{i,\mathsf{upr}})$. For any $i \in [\ell_n]$ and $\mathsf{tag} \in \{\mathsf{lwr}, \mathsf{mid}, \mathsf{upr}\}$, let $\mathsf{path}_{i,\mathsf{tag}}$ be a sequence of values $(\mathsf{node}_{i,1,\mathsf{tag}}, \mathsf{node}_{i,2,\mathsf{tag}}, \cdots, \mathsf{node}_{i,d_i,\mathsf{tag}})$[15]. $\mathcal{B}$ makes $\ell_n - \alpha + 1$ challenge encryption queries on (m_0, m_1) to the PKE challenger and obtains ciphertexts $\mathsf{ct}^*_\alpha, \mathsf{ct}^*_{\alpha+1}, \cdots, \mathsf{ct}^*_{\ell_n}$. For each $i < \alpha$, $\mathcal{B}$ sets $\tilde{y}_{i,d_i+1} = 1||\ \perp$. For each $i \geq \alpha$, $\mathcal{B}$ sets $\tilde{y}_{i,d_i+1} = 1||\mathsf{ct}^*_i$. For each $(i,j) \in S$, $\mathcal{B}$ simulates the garbling of $\mathsf{Enc\text{-}Step}_{i,j}$ as $(\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}, \tilde{y}_{i,j}) \leftarrow \mathsf{HG.Sim}(\mathsf{hk}, \mathsf{node}_{i,j}, 1^L, \mathsf{flag}_{i,j+1}||\tilde{y}_{i,j+1})$[13], where L is the length of circuit $\mathsf{Enc\text{-}Step}_{i,j}$ defined in Figure 9.2 and $\mathsf{flag}_{i,j+1} = 1$ iff $j = d_i$. Finally, $\mathcal{B}$ sends ciphertext $\mathsf{ct} = (\{(\mathsf{rt}_i, d_i)\}_{i\in[\ell_n]}, \{\widetilde{\mathsf{Enc\text{-}Step}}_{i,j}\}_{i\in[\ell_n], j\in[d_i]}, \{\tilde{y}_{i,1}\}_{i\in[\ell_n]})$ to $\mathcal{A}$. The adversary outputs a bit b'. $\mathcal{B}$ outputs b' as its guess in PKE game.

[17] For the ease of exposition, we consider the semantic security game where the adversary is allowed to make polynomially many challenge queries of the form (m_0, m_1). The challenger samples a bit b and responds to each challenge query (m_0, m_1) with $\mathsf{Enc}(\mathsf{pk}, m_b)$. By a standard hybrid argument, this definition of semantic security is equivalent to Definition 8.1.1.

We now analyze the advantage of $\mathcal{B}$ in breaking the semantic security of PKE. We note that if the PKE challenger encrypts the message m_0, then $\mathcal{B}$ emulates Hybrid $H_{0.\ell_n.d_{\ell_n}}$ challenger to $\mathcal{A}$. Whereas if the PKE challenger encrypts message m_1, then $\mathcal{B}$ emulates Hybrid $H_{1.\ell_n.d_{\ell_n}}$ challenger to $\mathcal{A}$. As $\mathcal{A}$ can distinguish between the hybrids with non-negligible probability, $\mathcal{B}$ can break PKE semantic security with non-negligible probability. ■

By the above sequence of lemmas and triangle inequality, Hybrid H_0 is computationally indistinguishable from Hybrid H_1. Therefore, our construction satisfies soundness for pre-registration verifiability property. ■

9.4.3 Message Hiding Security

We now prove that any VRBE construction that satisfies completeness and soundeness requirements of Definitions 8.3.2, 8.3.5 and 8.3.6 also satisfies message hiding property.

Theorem 9.4.3. *Assuming a VRBE satisfies completeness property (Definition 8.3.2) soundness of pre-registration and post-registration properties (Definitions 8.3.5 and 8.3.6), then the scheme also satisfies message hiding property (Definition 8.3.4).*

Proof. Consider the message hiding game described in Definition 8.3.4. We first divide the adversary into 2 types.

- Type 1 Adversary: Restricted to make challenge encryption query ($\mathsf{chal}, \mathsf{id}$, m_0, m_1) s.t. $\mathsf{id} = \mathsf{id}^*$, where id^* is the registered target identity.

- Type 2 Adversary: Restricted to make challenge encryption query ($\mathsf{chal}, \mathsf{id}$, m_0, m_1) s.t. $\mathsf{id} \notin S_{\mathsf{ID}}$, where S_{ID} is the set of all registered identities.

For any adversary $\mathcal{A}$, let the advantage of $\mathcal{A}$ in message hiding game be $\mathsf{Adv}^{\mathcal{A}}$. We now prove Lemmas 9.4.4 and 9.4.5 which together imply Theorem 9.4.3.

Lemma 9.4.4. *Assuming any VRBE satisfies completeness property as per Definition 8.3.2 and soundness of post-registration property as per Definition 8.3.6, then for every Type 1 PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\mathsf{Adv}^{\mathcal{A}} \leq negl(\lambda)$*

Proof. Suppose there exists a VRBE scheme VRBE satisfying completeness property as per Definition 8.3.2, a type 1 PPT adversary $\mathcal{A}$ such that the advantage $\mathsf{Adv}^{\mathcal{A}}$ is non-negligible. We now construct a PPT reduction algorithm $\mathcal{B}$ which breaks soundness of post-registration property of VRBE scheme with non-negligible probability.

The soundness game challenger $\mathcal{C}$ sends a $(\mathsf{crs}, \mathsf{pk}^*)$ to the reduction algorithm $\mathcal{B}$, which initializes parameters $(\mathsf{pp}, \mathsf{aux}, u, S_{\mathsf{ID}}, \mathsf{id}^*) = (\epsilon, \epsilon, \epsilon, \emptyset, \perp)$ and forwards crs to the adversary $\mathcal{A}$. The adversary then adaptively makes registration queries of the form $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$. For each such query, $\mathcal{B}$ aborts if $\mathsf{id} \in S_{\mathsf{ID}}$. Otherwise, $\mathcal{B}$ registers $(\mathsf{id}, \mathsf{pk})$ as $\mathsf{pp} \leftarrow \mathsf{Reg}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and sets $S_{\mathsf{ID}} = S_{\mathsf{ID}} \cup \{\mathsf{id}\}$. If the adversary makes a target registeration query $(\mathsf{regtgt}, \mathsf{id}^*)$, $\mathcal{B}$ aborts if $\mathsf{id}^* \in S_{\mathsf{ID}}$. Otherwise, $\mathcal{B}$ registers $(\mathsf{id}^*, \mathsf{pk}^*)$ as $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, sets $S_{\mathsf{ID}} = S_{\mathsf{ID}} \cup \{\mathsf{id}^*\}$ and sends pk^* to $\mathcal{A}$. Finally, the adversary makes a challenge query $(\mathsf{chal}, \mathsf{id}, m_0, m_1)$. If $\mathsf{id} \neq \mathsf{id}^*$, $\mathcal{B}$ aborts.

Otherwise, $\mathcal{B}$ computes proof $\pi^* \leftarrow \mathsf{PostProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$ and sends $(\mathsf{pp}, \mathsf{id}^*, \pi^*, m_0, m_1)$ to the challenger. If $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi^*) = 0$, the challenger aborts. Otherwise, it samples a bit $b \leftarrow \{0, 1\}$ and sends $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_b)$ to $\mathcal{B}$, which forwards ct^* to $\mathcal{A}$. The adversary $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs b' as its guess in soundness game.

We now analyze the advantage of $\mathcal{B}$ in the soundness game. As $\mathcal{A}$ is a valid type 1 adversary, it only makes challenge queries $(\mathsf{chal}, \mathsf{id}, m_0, m_1)$ s.t. $\mathsf{id} = \mathsf{id}^*$. As VRBE satisfies completeness property and $(\mathsf{id}^*, \mathsf{pk}^*)$ is registered by $\mathcal{B}$, we know that $\mathsf{PostVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*, \pi^*) = 1$ with overwhelming probability. Therefore, $\mathcal{B}$ does not abort and acts as a valid soundness game adversary and a valid message hiding game challenger with all but a negligible probability. As $\mathcal{A}$ has a non-negligible advantage in winning the message hiding game, $\mathcal{B}$ has a non-negligible advantage in winning the soundness game. ■

Lemma 9.4.5. *Assuming any VRBE satisfies completeness property as per Definition 8.3.2 and soundness of pre-registration property as per Definition 8.3.5, then for every Type 2 PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\mathsf{Adv}^{\mathcal{A}} \leq negl(\lambda)$*

Proof. This proof is similar to the proof Lemma 9.4.4. Suppose there exists a VRBE scheme VRBE satisfying completeness property as per Definition 8.3.2, a type 2 PPT adversary $\mathcal{A}$ such that the advantage $\mathsf{Adv}^{\mathcal{A}}$ is non-negligible. We now construct a PPT reduction algorithm $\mathcal{B}$, which breaks the soundness of the pre-registration property of the VRBE scheme with non-negligible probability.

The soundness game challenger $\mathcal{C}$ sends a crs to the reduction algorithm $\mathcal{B}$, which initializes parameters $(\mathsf{pp}, \mathsf{aux}, u, S_{\mathsf{ID}}, \mathsf{id}^*) = (\epsilon, \epsilon, \epsilon, \emptyset, \perp)$ and forwards crs to the adversary $\mathcal{A}$. The adversary then adaptively makes registration queries of the form $(\mathsf{regnew}, \mathsf{id}, \mathsf{pk})$. For each such query, $\mathcal{B}$ aborts if $\mathsf{id} \in S_{\mathsf{ID}}$. Otherwise, $\mathcal{B}$ registers $(\mathsf{id}, \mathsf{pk})$ as $\mathsf{pp} \leftarrow \mathsf{Reg}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ and sets $S_{\mathsf{ID}} = S_{\mathsf{ID}} \cup \{\mathsf{id}\}$. If the adversary makes a target registeration query $(\mathsf{regtgt}, \mathsf{id}^*)$, $\mathcal{B}$ aborts if $\mathsf{id}^* \in S_{\mathsf{ID}}$. Otherwise, $\mathcal{B}$ samples PKE pair $(\mathsf{pk}^*, \mathsf{sk}^*)$, registers $(\mathsf{id}^*, \mathsf{pk}^*)$ as $\mathsf{pp} \leftarrow \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, sets $S_{\mathsf{ID}} = S_{\mathsf{ID}} \cup \{\mathsf{id}^*\}$ and sends pk^* to $\mathcal{A}$. Finally, the adversary makes a challenge query $(\mathsf{chal}, \mathsf{id}, m_0, m_1)$. If $\mathsf{id} \in S_{\mathsf{ID}}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ computes proof $\pi \leftarrow \mathsf{PreProve}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id})$ and sends $(\mathsf{pp}, \mathsf{id}, \pi, m_0, m_1)$ to the challenger. If $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) = 0$, the challenger aborts. Otherwise, it samples a bit $b \leftarrow \{0, 1\}$ and sends $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m_b)$ to $\mathcal{B}$, which forwards ct^* to $\mathcal{A}$. The adversary $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs b' as its guess in soundness game.

We now analyze the advantage of $\mathcal{B}$ in the soundness game. As $\mathcal{A}$ is a valid type 2 adversary, it only makes challenge queries $(\mathsf{chal}, \mathsf{id}, m_0, m_1)$ s.t. $\mathsf{id} \notin S_{\mathsf{ID}}$. As VRBE satisfies completeness property and id is unregistered by $\mathcal{B}$, we know that $\mathsf{PreVerify}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \pi) = 1$ with overwhelming probability. Therefore, $\mathcal{B}$ does not abort and acts as a valid soundness game adversary and a valid message hiding game challenger with all but a negligible probability. As $\mathcal{A}$ has a non-negligible advantage in winning the message hiding game, $\mathcal{B}$ has a non-negligible advantage in winning the soundness game. ■

Appendix

Appendix 1

A Generic Construction of Hinting PRG

In this chapter, we generically construct (n, ℓ)-hinting PRG for any polynomials $n(\cdot), \ell(\cdot)$ from $(n-1, n, \ell)$-smooth recyclable OWFE, an extractor and a standard pseudorandom generator. Let $\mathcal{OWFE} = (K, f, E_1, E_2, D)$ be any $(n-1, n, \ell)$-smooth recyclable OWFE with randomness space $\mathcal{R}$, and let co-domain of f be $\mathcal{C}$. Let the min-entropy of distribution $\{f(x) : x \leftarrow \{0,1\}^n\}$ be k. As f is one-way, we know that $k = \Omega(\log \lambda)$. Let $\mathsf{Ext} : \mathcal{C} \times \mathcal{S} \to \mathcal{W}$ be a (k, ϵ) extractor, where ϵ is negligibly small in security parameter. Let $\mathsf{PRG} : \mathcal{W} \to \{0,1\}^\ell$ be a pseudorandom generator. We construct Hinting PRG with $(\mathsf{Setup}, \mathsf{Eval})$ as follows.

$\mathsf{Setup}(1^\lambda)$: First sample $\mathsf{pp}' \leftarrow K(1^\lambda)$. Then sample $\{\rho_{i,b}\}_{i\in[n],b\in\{0,1\}}$ uniformly at random from $\mathcal{R}$ and compute $\mathsf{ct}_{i,b} = E_1(\mathsf{pp}', (i,b); \rho_{i,b})$ for $i \in [n], b \in \{0,1\}$. Sample an extractor seed $\mathfrak{s} \leftarrow \mathcal{S}$. Output public parameters $\mathsf{pp} = (\mathsf{pp}', \{\mathsf{ct}_{i,b}\}_{i,b}, \mathfrak{s})$.

$\mathsf{Eval}(\mathsf{pp}, x, i)$: Parse pp as $(\mathsf{pp}', \{\mathsf{ct}_{i,b}\}_{i,b}, \mathfrak{s})$. If $i = 0$, output $\mathsf{PRG}(\mathsf{Ext}(f(\mathsf{pp}', x), \mathfrak{s}))$. Otherwise, output $D(\mathsf{ct}_{i,x_i}, x)$.

1.1 Security

We now prove that the above scheme is a secure hinting PRG. Formally, we prove

Theorem 1.1.1. *If $\mathcal{OWFE}$ is an $(n-1, n, \ell)$-smooth recyclable OWFE,* Ext *is a strong seeded extractor with appropriate parameters and* PRG *is a secure pseudorandom generator, then the above construction is a secure hinting PRG as per Definition 2.2.1.*

Proof. We prove the above theorem via a sequence of following hybrids. First, we modify the challenger to use E_2 algorithm to generate HPRG challenge. We then switch the randomly sampled $y_{i,1-x_i}$ values to be sampled in a structured way i.e., $y_{i,1-x_i} = E_2(\mathsf{pp}, (f(\mathsf{pp}, x), i, 1-x_i); \rho_{i,1-x_i})$. We then switch each of the challenge elements to random by using OWFE encryption security.

Hybrid H_0: This game corresponds to the original hinting prg game in which the challenger always chooses $\beta = 0$.

1. The challenger first samples OWFE public parameters $\mathsf{pp}' \leftarrow K(1^\lambda)$ and randomness $\{\rho_{i,b}\}_{i\in[n],b\in\{0,1\}}$. It then computes $\mathsf{ct}_{i,b} = E_1(\mathsf{pp}', (i,b); \rho_{i,b})$ for $i \in [n], b \in \{0,1\}$. The challenger then samples an extractor seed $\mathfrak{s}$.
2. The challenger samples a bit string $x \leftarrow \{0,1\}^n$, computes the challenge $y_0 = \mathsf{PRG}(\mathsf{Ext}(f(\mathsf{pp}', x), \mathfrak{s})), y_{i,x_i} = D(\mathsf{ct}_{i,x_i}, x), y_{i,\overline{x_i}} \leftarrow \{0,1\}^\ell$ for $i \in [n]$.

3. It sends public parameters $\mathsf{pp} = (\mathsf{pp}', \{ct_{i,b}\}_{i,b}, \mathfrak{s})$ and challenge $y = (y_0, \{y_{i,b}\}_{i,b})$ to the adversary.
4. The adversary outputs a bit β'.

Hybrid H_1: This is same as previous hybrid, except that the challenger computes y_{i,x_i} using E_2 algorithm.

2. The challenger samples a bit string $x \leftarrow \{0,1\}^n$, computes $t = f(\mathsf{pp}', x)$ and the challenge $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, $y_{i,x_i} = E_2(\mathsf{pp}', (t, i, x_i); \rho_{i,x_i})$, $y_{i,\overline{x_i}} \leftarrow \{0,1\}^\ell$ for $i \in [n]$.

We now define a sequence of $n+1$ hybrids. For the sake of simplicity, let Hybrid $H_{2.0}$ be same as Hybrid H_1.

Hybrid $H_{2.j}$ ($j \in [n]$): This is same as previous hybrid, except that the challenger computes $y_{i,\overline{x_i}}$ for $i \leq j$ differently.

2. The challenger samples a bit string $x \leftarrow \{0,1\}^n$, computes $t = f(\mathsf{pp}', x)$ and the challenge $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, $y_{i,b} = E_2(\mathsf{pp}', (t, i, b); \rho_{i,b})$ for all (i, b) s.t. $i \leq j$ or $b = x_i$ and samples $y_{i,1-x_i} \leftarrow \{0,1\}^\ell$ for all $i > j$.

We now define a sequence of $2n+1$ hybrids. For the sake of simplicity, let Hybrid $H_{3.0.1}$ be same as Hybrid $H_{2.n}$.

Hybrid $H_{3.j.b'}$ ($j \in [n], b' \in \{0,1\}$): In this hybrid, the challenger samples x s.t. $x_j = 1 - b'$. It also samples $y_{i,b}$ uniformly at random for all $(i, b) \prec (j, b')$.

2. The challenger samples a bit string $x \leftarrow \{0,1\}^n$ s.t. $x_j = 1 - b'$, computes $t = f(\mathsf{pp}', x)$ and the challenge $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, computes $y_{i,b} = E_2(\mathsf{pp}', (t, i, b); \rho_{i,b})$ for $(i, b) \succeq (j, b')$ and samples $y_{i,b} \leftarrow \{0,1\}^\ell$ for $(i, b) \prec (j, b')$.

We now define a sequence of $2n$ hybrids.

Hybrid $H_{4.j.b'}$: This hybrid is same as Hybrid $H_{3.j.b'}$, except that the challenger samples $y_{j,b'}$ at random.

2. The challenger samples a bit string $x \leftarrow \{0,1\}^n$ s.t. $x_j = 1 - b'$, computes $t = f(\mathsf{pp}', x)$ and the challenge $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, computes $y_{i,b} = E_2(\mathsf{pp}', (t, i, b); \rho_{i,b})$ for $(i, b) \succ (j, b')$ and samples $y_{i,b} \leftarrow \{0,1\}^\ell$ for $(i, b) \preceq (j, b')$.

Hybrid H_5: This hybrid is same as Hybrid $H_{4.n.1}$, except that the challenger samples $x \leftarrow \{0,1\}^n$ without any restriction.

2. The challenger samples a bit string $x \leftarrow \{0,1\}^n$, computes $t = f(\mathsf{pp}', x)$ and the challenge $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, $y_{i,b} \leftarrow \{0,1\}^\ell$ for all $i \in [n], b \in \{0,1\}$.

Hybrid H_6: This is same as Hybrid $H_{5.n.1}$, except that the challenger samples t uniformly at random.

2. The challenger samples $t \leftarrow \mathcal{W}$, computes $y_0 = \mathsf{PRG}(t)$, $y_{i,b} \leftarrow \{0,1\}^\ell$ for all $i \in [n], b \in \{0,1\}$.

Hybrid H_7: This is same as Hybrid H_6, except that the challenger samples y_0 uniformly at random.

2. The challenger samples $y_0 \leftarrow \{0,1\}^\ell$, $y_{i,b} \leftarrow \{0,1\}^\ell$ for all $i \in [n], b \in \{0,1\}$.

Note that hybrid H_0 is the original hinting PRG game when challenger always chooses $\beta = 0$ and hybrid H_7 is the original hinting PRG game when challenger always chooses $\beta = 1$. We prove that these 2 hybrids are computationally indistinguishable using the following lemmas. For any PPT adversary $\mathcal{A}$, let $p_s^{\mathcal{A}}$ be the probability that $\mathcal{A}$ outputs 1 in Hybrid H_s.

Lemma 1.1.1. *Assuming $\mathcal{OWFE}$ is perfectly correct, for any adversary $\mathcal{A}$, we have $p_0^{\mathcal{A}} = p_1^{\mathcal{A}}$.*

Proof. Assuming $\mathcal{OWFE}$ is perfectly correct, the distribution of the challenger's output is same in hybrids H_0 and H_1. ■

Lemma 1.1.2. *Assuming $\mathcal{OWFE}$ has secure encryption property, for any PPT adversary $\mathcal{A}$, and index $j \in [n]$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p_{2.j}^{\mathcal{A}} - p_{2.j-1}^{\mathcal{A}}| \leq negl(\lambda)$.*

Proof. Suppose there exists a PPT Adversary $\mathcal{A}$, and an index $j \in [n]$ such that $|p_{2.j}^{\mathcal{A}} - p_{2.j-1}^{\mathcal{A}}|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that breaks OWFE security of encryption.

The reduction algorithm first samples $x \leftarrow \{0,1\}^n$ and sends (x, j) to challenger $\mathcal{C}$. The challenger samples OWFE public parameters $\mathsf{pp}' \leftarrow K(1^\lambda)$,

a bit $\alpha \leftarrow \{0,1\}$, randomness ρ and computes $\mathsf{ct}^* = E_1(\mathsf{pp}', j, 1-x_j; \rho)$. If $\alpha = 1$, it computes $y^* = E_2(\mathsf{pp}', f(\mathsf{pp}', x), j, 1-x_j; \rho)$. Otherwise, it samples $y^* \leftarrow \{0,1\}^\ell$. The challenger sends $(\mathsf{pp}', \mathsf{ct}^*, y^*)$ to $\mathcal{B}$. $\mathcal{B}$ then samples randomness $\rho_{i,b} \leftarrow \mathcal{R}$ and computes $\mathsf{ct}_{i,b} = E_1(\mathsf{pp}', (i,b); \rho_{i,b})$ for $(i,b) \neq (j, 1-x_j)$. It then initializes $\mathsf{ct}_{j,1-x_j} = \mathsf{ct}^*$, $y_{j,1-x_j} = y^*$. $\mathcal{B}$ then samples an extractor seed $\mathfrak{s} \leftarrow \mathcal{S}$, computes $t = f(\mathsf{pp}', x)$, $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, and $y_{i,b} = E_2(\mathsf{pp}', (t, i, b); \rho_{i,b})$ for all (i,b) s.t. $i < j \vee b = x_i$. It then samples $y_{i,1-x_i} \leftarrow \{0,1\}^\ell$ for $i > j$ and sends public parameters $\mathsf{pp} = (\mathsf{pp}', \{ct_{i,b}\}_{i,b}, \mathfrak{s})$ and challenge $(y_0, \{y_{i,b}\}_{i,b})$ to the adversary $\mathcal{A}$. The adversary outputs a bit β'. The reduction algorithm $\mathcal{B}$ outputs β' as its guess in OWFE game.

Note that if $\alpha = 0$, $\mathcal{B}$ emulates Hybrid $H_{2.j-1}$ challenger to $\mathcal{A}$. If $\alpha = 1$, $\mathcal{B}$ emulates Hybrid $H_{2.j}$ challenger to $\mathcal{A}$. Moreover, $\mathcal{B}$ acts as a valid adversary in OWFE encryption security game. By our assumption, $|p^{\mathcal{A}}_{2.j} - p^{\mathcal{A}}_{2.j-1}| = |\Pr[\beta' = 1|\alpha = 0] - \Pr[\beta' = 1|\alpha = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks OWFE encryption security. ■

Lemma 1.1.3. *Assuming $(n-1, n)$-smoothness of $\mathcal{OWFE}$ scheme, for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{2.n} - p^{\mathcal{A}}_{3.1.0}| \leq negl(\lambda)$.*

Proof. Suppose there exists a PPT Adversary $\mathcal{A}$ such that $|p^{\mathcal{A}}_{2.n} - p^{\mathcal{A}}_{3.1.0}|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that breaks smoothness property of $\mathcal{OWFE}$.

Let $X_0 = \{0,1\}^n$ and $X_1 = \{x \in \{0,1\}^n : x_1 = 1\}$. Let S_0 and S_1 be uniform distributions on sets X_0 and X_1 respectively. Note that both S_0 and S_1 have min-entropy at least $n-1$. The reduction algorithm $\mathcal{B}$ sends S_0 and S_1 to the challenger $\mathcal{C}$ of smoothness game. The challenger $\mathcal{C}$ samples a bit $\alpha \leftarrow \{0,1\}$, samples $x \leftarrow S_\alpha$, samples OWFE public parameters $\mathsf{pp}' \leftarrow K(1^\lambda)$, and sends $(\mathsf{pp}', t = f(\mathsf{pp}', x))$ to $\mathcal{B}$. The reduction algorithm $\mathcal{B}$ samples an extractor seed $\mathfrak{s}$, randomness $\rho_{i,b} \leftarrow \mathcal{R}$, computes $\mathsf{ct}_{i,b} = E_1(\mathsf{pp}', (i,b); \rho_{i,b})$, $y_{i,b} = E_2(\mathsf{pp}', (t,i,b); \rho_{i,b})$ for $i \in [n], b \in \{0,1\}$ and sets $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$. $\mathcal{B}$ sends public parameters $\mathsf{pp} = (\mathsf{pp}', \{\mathsf{ct}_{i,b}\}_{i,b}, \mathfrak{s})$ and challenge $(y_0, \{y_{i,b}\}_{i,b})$ to $\mathcal{A}$. The adversary outputs a bit β', which $\mathcal{B}$ outputs as its guess in smoothness game.

Note that if $\alpha = 0$, $\mathcal{B}$ emulates Hybrid $H_{2.n}$ challenger to $\mathcal{A}$. If $\alpha = 1$, $\mathcal{B}$ emulates Hybrid $H_{3.1.0}$ challenger to $\mathcal{A}$. Moreover, $\mathcal{B}$ acts as a valid adversary in OWFE encryption security game. By our assumption, $|p^{\mathcal{A}}_{2.n} - p^{\mathcal{A}}_{3.1.0}| = |\Pr[\beta' = 1 | \alpha = 0] - \Pr[\beta' = 1 | \alpha = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks $(n-1, n)$-smoothness property of $\mathcal{OWFE}$. ■

Lemma 1.1.4. *Assuming $\mathcal{OWFE}$ has secure encryption property, for any PPT adversary $\mathcal{A}$, index $j \in [n]$, bit $b' \in \{0,1\}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{3.j.b'} - p^{\mathcal{A}}_{4.j.b'}| \leq negl(\lambda)$.*

Proof. Suppose there exists a PPT Adversary $\mathcal{A}$, an index $j \in [n]$, bit $b' \in \{0,1\}$ such that $|p^{\mathcal{A}}_{3.j.b'} - p^{\mathcal{A}}_{4.j.b'}|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that breaks OWFE security of encryption.

The reduction algorithm first samples $x \leftarrow \{0,1\}^n$ s.t. $x_j = 1 - b'$, and sends (x, j) to challenger $\mathcal{C}$. The challenger samples OWFE public parameters $\mathsf{pp}' \leftarrow K(1^\lambda)$, a bit $\alpha \leftarrow \{0,1\}$, randomness ρ and computes $\mathsf{ct}^* = E_1(\mathsf{pp}', j, 1 - x_j; \rho)$. If $\alpha = 0$, it computes $y^* = E_2(\mathsf{pp}', f(\mathsf{pp}', x), j, 1 - x_j; \rho)$. Otherwise, it samples $y^* \leftarrow \{0,1\}^\ell$. The challenger sends $(\mathsf{pp}', \mathsf{ct}^*, y^*)$ to $\mathcal{B}$. $\mathcal{B}$ then samples randomness $\rho_{i,b} \leftarrow \mathcal{R}$ and computes $\mathsf{ct}_{i,b} = E_1(\mathsf{pp}', (i,b); \rho_{i,b})$ for $(i,b) \neq (j,b')$. It then initializes $\mathsf{ct}_{j,b'} = \mathsf{ct}^*$, $y_{j,b'} = y^*$. $\mathcal{B}$ then computes $t = f(\mathsf{pp}', x)$, $y_0 = \mathsf{PRG}(\mathsf{Ext}(t, \mathfrak{s}))$, and $y_{i,b} = E_2(\mathsf{pp}', (t, i, b); \rho_{i,b})$ for all (i,b) s.t. $(i,b) \succ (j,b')$. It then samples $y_{i,b} \leftarrow \{0,1\}^\ell$ for all $(i,b) \prec (j,b')$ and sends public parameters $\mathsf{pp} = (\mathsf{pp}', \{ct_{i,b}\}_{i,b}, \mathfrak{s})$ and challenge $(y_0, \{y_{i,b}\}_{i,b})$ to the adversary $\mathcal{A}$. The adversary outputs a bit β'. The reduction algorithm $\mathcal{B}$ outputs β' as its guess in OWFE game.

Note that if $\alpha = 0$, $\mathcal{B}$ emulates Hybrid $H_{3.j.b'}$ challenger to $\mathcal{A}$. If $\alpha = 1$, $\mathcal{B}$ emulates Hybrid $H_{3.j.b'}$ challenger to $\mathcal{A}$. Moreover, $\mathcal{B}$ acts as a valid adversary in OWFE encryption security game. By our assumption, $|p^{\mathcal{A}}_{3.j.b'} - p^{\mathcal{A}}_{4.j.b'}| = |\Pr[\beta' = 1 | \alpha = 0] - \Pr[\beta' = 1 | \alpha = 1]|$ is non-negligible. Therefore, $\mathcal{B}$ breaks OWFE encryption security. ■

Lemma 1.1.5. *Assuming $(n-1, n)$-smoothness of $\mathcal{OWFE}$ scheme, for any PPT adversary $\mathcal{A}$, index $j \in [n]$, bit $b' \in \{0,1\}$, s.t. $(j, b') \prec (n, 1)$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{4.j.b'} - p^{\mathcal{A}}_{3.j+b'.1-b'}| \leq negl(\lambda)$.*

Proof. This proof is similar to proof of Lemma 1.1.3.

■

Lemma 1.1.6. *Assuming $(n-1,n)$-smoothness of $\mathcal{OWFE}$ scheme, for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_{4.n.1} - p^{\mathcal{A}}_5| \leq negl(\lambda)$.*

Proof. This proof is similar to proof of Lemma 1.1.3. ■

Lemma 1.1.7. *Assuming $\mathcal{OWFE}$ satisfies one-wayness property and Ext is a strong seeded extractor with appropriate parameters, for any adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_5 - p^{\mathcal{A}}_6| \leq negl(\lambda)$.*

Proof. Assuming f is one-way, we know that the min-entropy k of the distribution $\{f(x) : x \leftarrow \{0,1\}^n\}$ is $\Omega(\log \lambda)$ bits. If $\mathsf{Ext} : \mathcal{C} \times \mathcal{S} \rightarrow \mathcal{W}$ is a strong seeded (k, ϵ) extractor for a negligibly small ϵ, then the distributions $(\mathfrak{s}, \mathsf{Ext}(f(\mathsf{pp}', x)))$ and $(\mathfrak{s}, u)$, where $\mathfrak{s} \leftarrow \mathcal{S}, \mathsf{pp}' \leftarrow K(1^\lambda), x \leftarrow \{0,1\}^n, w \leftarrow \mathcal{W}$, have a statistical difference of ϵ. Moroever, with an appropriate choice of ϵ, we have $|\mathcal{W}| \geq 2^{\Omega(\log \lambda)}$. ■

Lemma 1.1.8. *Assuming $\mathsf{PRG} : \mathcal{W} \rightarrow \{0,1\}^\ell$ is a secure PRG for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have $|p^{\mathcal{A}}_6 - p^{\mathcal{A}}_7| \leq negl(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $|p^{\mathcal{A}}_6 - p^{\mathcal{A}}_7|$ is non-negligible. We construct a reduction algorithm $\mathcal{B}$ that breaks PRG security.

The PRG challenger $\mathcal{C}$ samples a bit $\alpha \leftarrow \{0,1\}$. If $\alpha = 0$, it samples $t \leftarrow \mathcal{W}$ and lets $v = \mathsf{PRG}(t)$. Otherwise, it samples $v \leftarrow \{0,1\}^\ell$. The challenger sends v to the reduction algorithm $\mathcal{B}$, which samples HPRG public parameters pp, $y_{i,b} \leftarrow \{0,1\}^\ell$ for $i \in [n], b \in \{0,1\}$, sets $y_0 = v$ and sends pp, challenge $(y_0, \{y_{i,b}\}_{i,b})$ to $\mathcal{A}$. The adversary outputs a bit β'. $\mathcal{B}$ outputs β' as its guess in PRG game.

Note that, $\mathcal{B}$ acts as Hybrid H_6 challenger if $\alpha = 0$, and as Hybrid H_7 challenger if $\alpha = 1$. By our assumption, $|p_6^{\mathcal{A}} - p_7^{\mathcal{A}}| = |\Pr[\beta' = 1|\alpha = 0] - \Pr[\beta' = 1|\alpha = 1]|$ is non-negligible and $\mathcal{B}$ breaks PRG security. ■

By the above sequence of lemmas and triangle inequality, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_0^{\mathcal{A}} - p_7^{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$. Therefore, the above construction is a secure hinting prg. ■

Bibliography

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[2] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security*, pages 452–473. Springer, 2003.

[3] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In *CT-RSA 2009*, 2009.

[4] M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks -BRSIM/UC-soundness of Dolev-Yao-style encryption with key cycles. *J.of Comp.Security*, (5), 2008.

[5] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, 2001.

[6] Niko Baric and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97*, 1997.

[7] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC 2005*, 2005.

[8] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, 1993.

[9] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In *EUROCRYPT '93*, 1993.

[10] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *EUROCRYPT*, 2018.

[11] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multiprover interactive proofs and their efficiency benefits. In *Advances in Cryptology - CRYPTO 2012*, pages 255–272, 2012.

[12] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC 2002*, 2002.

[13] Dan Boneh and Xavier Boyen. Efficient selective-ID secure Identity-Based Encryption without random oracles. In *EUROCRYPT '04*, 2004.

[14] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT '04*, 2004.

[15] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *CRYPTO '01*, 2001.

[16] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-Secure Encryption from Decision Diffie-Hellman. In *CRYPTO '08*, 2008.

[17] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *TCC'11*, Berlin, Heidelberg, 2011. Springer-Verlag.

[18] Xavier Boyen and Brent Waters. Shrinking the keys of discrete-log-type lossy trapdoor functions. In *ACNS*, 2010.

[19] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO 2010*, 2010.

[20] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018*.

[21] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. Cryptology ePrint Archive, Report 2017/967, 2017. `http://eprint.iacr.org/2017/967`.

[22] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT '99*, 1999.

[23] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC 2009*, 2009.

[24] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, 2002.

[25] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from Identity Based Encryption. In *EUROCRYPT '04*, 2004.

[26] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, 2010.

[27] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *PKC 2013*, 2013.

[28] Liqun Chen, Keith Harrison, David Soldera, and Nigel P Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *International Conference on Infrastructure Security*, pages 260–275. Springer, 2002.

[29] Zhaohui Cheng, Richard Comley, and Luminita Vasiu. Remove key escrow from the identity-based encryption system. In *Exploring New Frontiers of Theoretical Informatics*, pages 37–50. Springer, 2004.

[30] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *CRYPTO 2017*, 2017.

[31] Sherman SM Chow. Removing escrow from identity-based encryption. In *International Workshop on Public Key Cryptography*, pages 256–276. Springer, 2009.

[32] Clifford Cocks. An identity based encryption scheme based on Quadratic Residues. In *Cryptography and Coding, IMA International Conference*, 2001.

[33] Charles-Jean De la Vallée Poussin. *Recherches analytiques sur la théorie des nombres premiers*. Hayez, Imprimeur de l'Académie royale de Belgique, 1897.

[34] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6), 1976.

[35] PG Lejeune Dirichlet. Beweis eines satzes über die arithmetische progression. *Bericht über die Verhandlungen der königlich Presussischen Akademie der Wissenschaften Berlin*, 1837.

[36] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, (1), 2008.

[37] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*, 2004.

[38] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *CRYPTO 2017*, 2017.

[39] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Kevin Liu, and Giulio Malavolta. Rate-1 trapdoor functions from the diffie-hellman problem. In *ASIACRYPT*, 2019.

[40] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In *PKC 2018*, 2018.

[41] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019*, 2019.

[42] Nico Döttling and Sanjam Garg. From selective ibe to full ibe and selective hibe. TCC, 2017.

[43] Georgios Fotiadis and Elisavet Konstantinou. TNFS resistant families of pairing-friendly elliptic curves. *IACR Cryptology ePrint Archive*, 2018.

[44] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In *EUROCRYPT 2019*, 2019.

[45] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[46] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational diffie-hellman assumption. In *CRYPTO 2018*, 2018.

[47] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In *TCC 2018*, 2018.

[48] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In *PKC 2019*, 2019.

[49] Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 ot and more. Cryptology ePrint Archive, Report 2019/990, 2019. `https://eprint.iacr.org/2019/990`.

[50] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In *FOCS 2017*, 2017.

[51] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT*, 2018.

[52] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT '06*, 2006.

[53] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[54] Craig Gentry and Zulfikar Ramzan. Rsa accumulator based broadcast encryption. In *International Conference on Information Security*. Springer, 2004.

[55] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011*, pages 99–108, 2011.

[56] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 1984.

[57] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *STOC*. ACM, 1985.

[58] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020*.

[59] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. New constructions of hinting prgs, owfs with encryption, and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020*, 2020.

[60] Vipul Goyal. Reducing trust in the PKG in identity based cryptosystems. In *CRYPTO '07*, 2007.

[61] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 427–436. ACM, 2008.

[62] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, 2006.

[63] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, 2010.

[64] Shai Halevi and Hugo Krawczyk. Security under key-dependent inputs. In *ACM CCS '07*, 2007.

[65] W. B. Hart. Fast library for number theory: An introduction. In *Proceedings of the Third International Congress on Mathematical Software*, ICMS'10, Berlin, Heidelberg, 2010. Springer-Verlag. `http://flintlib.org`.

[66] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, (4), 1999.

[67] Brett Hemenway, Rafail Ostrovsky, and Alon Rosen. Non-committing encryption from Φ-hiding. In *TCC 2015*, 2015.

[68] Herumi. A portable and fast pairing-based cryptography library. `https://github.com/herumi/mcl`, 2019.

[69] Dennis Hofheinz and Dominique Unruh. Towards key-dependent message security in the standard model. In *EUROCRYPT '08*, 2008.

[70] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO 2009*, 2009.

[71] Aniket Kate and Ian Goldberg. Distributed private-key generators for identity-based cryptography. In *International Conference on Security and Cryptography for Networks*, pages 436–453. Springer, 2010.

[72] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing nizks from diffie-hellman assumptions. In *EUROCRYPT 2019*, 2019.

[73] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *CRYPTO 2016*, 2016.

[74] Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. Cca security and trapdoor functions via key-dependent-message security. In *Crypto '19*, 2019.

[75] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *CRYPTO 2019*, 2019.

[76] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Secure key issuing in id-based cryptography. In *ACSW Frontiers 2004, 2004 ACSW Workshops*, pages 69–74, 2004.

[77] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, 2010.

[78] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012*, pages 169–189, 2012.

[79] Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier nizks. In *EUROCRYPT 2019*, 2019.

[80] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. In *Mycrypt 2016*, 2016.

[81] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science*, pages 436–453, 1994.

[82] Moni Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology - CRYPTO 2003*, pages 96–109, 2003.

[83] Donald J Newman. Simple analytic proof of the prime number theorem. *The American Mathematical Monthly*, (9), 1980.

[84] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology - CT-RSA 2005*, 2005.

[85] Kenneth G Paterson and Sriramkrishnan Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In *International Conference on Pairing-Based Cryptography*, pages 354–375. Springer, 2008.

[86] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier nizks for all NP from CDH. In *EUROCRYPT 2019*, 2019.

[87] Michael Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[88] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, (2), 1978.

[89] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. https://eprint.iacr.org/2015/1162.

[90] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[91] Amit Sahai and Brent Waters. Slides on functional encryption. PowerPoint presentation, 2008. http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt.

[92] Tomas Sander, Amnon Ta-Shma, and Moti Yung. Blind, auditable membership proofs. In *FC 2000*, 2000.

[93] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst.*, (1), 1983.

[94] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO '84*, 1984.

[95] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 84*, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[96] Ivan Soprounov. A short proof of the prime number theorem for arithmetic progressions. *preprint*, 2010.

[97] Terence Tao. The prime number theorem in arithmetic progressions, and dueling conspiracies. Terry Tao Blog Post, 24 September, 2009. https://terrytao.wordpress.com/2009/09/24/the-prime-number-theorem-in-arithmetic-

[98] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, 2005.

[99] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, 2009.

[100] Quanyun Wei, Fang Qi, and Zhe Tang. Remove key escrow from the BF and gentry identity-based encryption with non-interactive key generation. *Telecommunication Systems*, 69(2):253–262, 2018.

[101] Andrew Yao. How to generate and exchange secrets. In *FOCS*, 1986.

[102] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*. IEEE Computer Society, 1982.

[103] Don Zagier. Newman's short proof of the prime number theorem. *The American mathematical monthly*, (8), 1997.

[104] Mark Zhandry. The magic of elfs. In *CRYPTO 2016*, 2016.

Index