Privately Constrained Testable Pseudorandom Functions

by

Filip Pawlega

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Privately Constrained Pseudorandom Functions allow a PRF key to be delegated to some evaluator in a constrained manner, such that the key's functionality is restricted with respect to some secret predicate. Variants of Privately Constrained Pseudorandom Functions have been applied to rich applications such as Broadcast Encryption, and Secret-key Functional Encryption. Recently, this primitive has also been instantiated from standard assumptions. We extend its functionality to a new tool we call Privately Constrained *Testable* Pseudorandom functions.

For any predicate C, the holder of a secret key sk can produce a delegatable key constrained on C denoted as $\mathsf{sk}[C]$. Evaluations on inputs x produced using the constrained key differ from unconstrained evaluations with respect to the result of $C(x)$. Given an output y evaluated using $\mathsf{sk}[C]$, the holder of the unconstrained key sk can verify whether the *input* x used to produce y satisfied the predicate C. That is, given y, they learn whether $C(x) = 1$ without needing to evaluate the predicate themselves, and without requiring the original input x.

We define two inequivalent security models for this new primitive, a stronger indistinguishability-based definition, and a weaker simulation-based definition. Under the indistinguishability-based definition, we show the new primitive implies Designated-Verifier Non-Interactive Zero-Knowledge Arguments for NP in a black-box manner. Under the simulation-based definition, we construct a provably secure instantiation of the primitive from lattice assumptions. We leave the study of the gap between definitions, and discovering techniques to reconcile it as future work.

Acknowledgements

Firstly, I thank my advisors Sergey Gorbunov and David Jao for their patience, support, and inspiration. For introducing me to countless ideas — and advice on both life and technical matters.

To my readers Florian Kerschbaum and Alfred Menezes for their attention, corrections, and helpful suggestions.

To my colleagues in CrySP, and at ISARA: for being passionate about their work and motivating me to do the same.

To Adam, Katerina, and Alon, for many enlightening and collectively perplexing discussions — and to Hoeteck, for telling me about hi and lo-tech techniques.

To my friends Andrew, Dhinakaran, Luis, and Neil for listening to me ramble on countless occasions.

To my family, and all those who helped me reach here.

Most importantly, to Meagan, whose love made this possible.

Table of Contents

List of Figures vii

1 Introduction 1

1.1 Thesis Outline 4

2 Pseudorandomness and Background 5

2.1 Notation and Cryptographic Background 5

2.2 Pseudorandom Functions 7

3 Range-testable Pseudorandom Functions 13

3.1 Definitions 13

4 Applications 17

4.1 Designated-verifier Non-interactive Zero-knowledge 17

4.1.1 Security 21

4.1.2 Comparison with other Works 25

5 Lattice-based Cryptography 27

5.1 Lattices and Computational Problems 28

5.2 Cryptographic Assumptions 30

5.3 Lattice Trapdoor Techniques 34

6 Construction from Learning with Errors **37**
6.1 Lattice-based PRF Fundamentals 37
6.2 GGH15 Encodings and Branching Programs 39
6.3 Construction 48
6.3.1 Construction Overview 48
6.3.2 Correctness and Parameters 51
6.3.3 Security 54

7 Extensions and Discussions **59**
7.1 Conditional One-wayness 59
7.2 Context-Hiding 60
7.2.1 Instantiating Context-Hiding 61
7.3 Feasibility of the NIZK Application 63
7.4 Testability of other PC-PRFs 63
7.5 Relation to other GGH15 Constructions 64
7.6 One-sided Testability 66

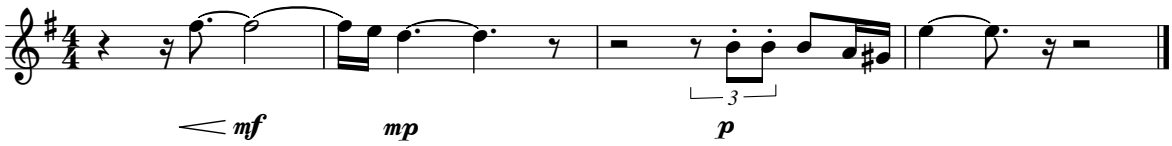
8 Conclusions and Future Work **68**

References **70**

List of Figures

4.1 Constraint circuit C_i 20

6.1 A simple directed graph 39



- Miles Davis, *Blue in Green*

Chapter 1

Introduction

In recent years, the landscape of theoretical cryptography has expanded considerably from a wealth of new ideas. Nevertheless, long-standing challenging problems resist satisfying solutions and drive a search for deeper insights and subtle relationships between concepts. Definitions are often revisited and refined, proofs of creative techniques remain elusive for years, and new impossibility results are frequent. In the past, we were restricted to securing coarse-grained access to data. Today, we have techniques for computing on data while it remains encrypted, and other fine-grained expressive tools. More recently, extensive research has been done quantifying the requirements and necessary security limitations of not only computing over encrypted data, but the evaluation of secret functions.

Consider a user Alice who possesses some private function f, and a user Bob who possesses some inputs $(x_1, \ldots, x_k)$. Alice wishes to learn $(f(x_1), \ldots, f(x_k))$ without requiring Bob's inputs, and without revealing f to Bob. Tools in the space of multi-party computation have been able to solve this problem for some time, though with limitations. One of the earliest schemes in this area is *Garbled Circuits*, due to Yao. Garbled Circuits (GC) allow for *single-use* evaluation of a circuit C on an input x such that the only information learned by an evaluator is $C(x)$, and the length of the circuit. Reusability was only achieved years later by Goldwasser et al. [38]. Additionally, this approach requires Bob's inputs to first be encoded in some way by Alice. Since we would like Alice's behaviour to be independent of Bob's inputs, this doesn't quite work. Though, we could at least allow Alice to perform the encoding without learning the inputs, by introducing another primitive called Oblivious Transfer.

A more general tool called *Functional Encryption* (FE) allows for Bob to encrypt his inputs $\{x_i\}$ himself using a public key, and be given a *functional decryption key* which only

outputs decryptions of $\{f(x_i)\}$. However, achieving function-privacy in this setting quickly takes on the flavour of a much more powerful primitive called Indistinguishability Obfuscation (iO). Indistinguishability Obfuscation guarantees that any two functions f_0, f_1 which have identical input and output distributions can be encoded in a manner which preserves functionality but hides which function is encoded. This seemingly restrictive capability was show to be quite powerful by Sahai and Waters ([57]), who demonstrated techniques for obtaining Deniable Encryption, Compact Signatures, and many other interesting applications. Recently, new candidate iO schemes have been proven secure without the use of multi-linear maps — although requiring relying on new assumptions ([7]), or additional tools which are yet to be instantiated ([2]).

In order to move away from the realm of obfuscation, one idea to consider is what happens if Alice gives Bob some manner of reusable encoding of f which hides its description, but also requires secret information in order to interpret the outputs. Then, perhaps we can create a primitive in which Bob can produce encodings of $\{f(x_i)\}$ for arbitrary inputs. Then, we can ask along which dimensions we can explore trade-offs in feasability and security for such a primitive?

One existing tool which has a similar flavour to this idea is *Privately Verifiable Computation*. That primitive allows Alice to delegate a computation to Bob, and can be extended to support function and output privacy. Though, in that case Alice must provide the encoding of the input herself. Or, perhaps we could try to use Fully Homomorphic Encryption (FHE) to encrypt the circuit representation of f. Then, Bob could choose an input x and compute the encryption of the universal circuit $\mathcal{U}_x$. This would allow him to compute encryptions of $f(x)$ without learning f by computing a homomorphic evaluation taking $\mathsf{Enc}_{pk}(f)$ as input. Unfortunately, the malleability of FHE means Bob could easily change the circuit description, or possibly the output as well. Again, we don't get quite what we would like.

So, we turn our attention to *Privately Constrained* Pseudorandom Functions (PC-PRFs). Constrained PRFs were concurrently introduced in [18], [46], and [19]. Intuitively, this tool allows the holder of a PRF delegate a restricted version of their key which allows evaluation for a subset of the domain. For constrained, or "unauthorized" inputs, evaluations performed using the constrained key differ from the function outputs produced by the original key, otherwise the output distributions should be identical for the remainder of the domain. Without the restriction, we can make no security arguments with respect to the pseudorandomness property of PRFs since the holder of an unconstrained key has trivial distinguishing power. Varying degrees of expressiveness in the constraint can be used to yield primitives such as Broadcast Encryption and Non-Interactive Identity-Based Key Exchange.

Our starting intuition is that for regular *circuit-constrained* PRFs, evaluation under a constrained key in fact implicitly produces an *encoding* of a function evaluation in the form of a PRF output. Consider a constraint predicate C and an associated constrained PRF key $\mathsf{sk}[C]$. Then, the output of a constrained evaluation on an input x (computed using $\mathsf{sk}[C]$) is a deterministic pseudorandom value y. Moreover, y is equal to an unconstrained evaluation on x computed using sk if and only if $C(x) = 1$. Hence, whether or not the equality is satisfied encodes a single bit. However, the definition does not require the constraint to be private, so nothing is hidden from Bob.

A subsequent work by Boneh et al. added constraint-privacy to the definition [17]. However, properly characterizing security of the primitive proved to be non-trivial. In a followup work, [14] it was shown that a candidate simulation-based security definition was not satisfiable. Soon after, Canetti and Chen ([27]) showed the original definition must be restricted to handing out a single constrained key, as security for multiple keys would immediately imply iO. They also proposed a weaker simulation definition which could actually be realized. They also showed that PC-PRFs imply Secret-key Functional Encryption. Since then, several constructions of PC-PRFs have been shown under this definition based on lattice assumptions.

Using PC-PRFs, the intuition sketched above is a good starting point, but not enough to satisfy our goal. While constraint-privacy allows Alice to decode a bit privately, she can do this only knowing Bob's input if her test is an equality comparison. We would like Alice's test to be independent of Bob's input. Additionally, it would be nice if Alice's computational effort did scale with the size of the hidden constraint. To accomplish this, we introduce a Privately Constrained *Testable* PRF (PCT-PRF). This new tool inherits the properties of PC-PRFs, but allows Alice to check whether a PRF output resides in a range corresponding to $b = C(x)$ without requiring x. Alice recovers the bit b through the implication that the output must have been produced by some $x : C(x) = b$. In applications, our requirements will be somewhat strengthened to allow reusability and avoid knowledge assumptions.

We will define the properties of this new primitive, and show how to construct it from lattice assumptions. The construction will follow by observing that several techniques from recent works for encoding *branching programs* (which build upon the result of Gorbunov et al. in [34]) can be combined together. This lets us first construct a privately constrained PRF with a particular structure, and then apply standard techniques to extend it to admit testability in a straightforward manner. Similar observations regarding recent extensions of GGH15 were made independently (but not concurrently) in the work by Chen et al. ([31] and a later update to [27]), such that the resulting construction is almost identical to that found in their work. However, they do not define or consider testability.

We will also show how to use the Privately Constrained Testable PRF to build a Non-Interactive Zero-Knowledge (NIZK) proof system for NP, and prove it is secure with respect to a candidate security definition for the testable PRF. Unfortunately, as in other PC-PRFs, correctly stating security is challenging and we are only be able to prove *the testable PRF* itself is secure with respect to a weaker security definition. In particular, the NIZK security analysis does not carry over to the weaker definition. We conclude by leaving this gap, and related technical challenges as open problems for future study. Recently, Kim and Wu [47] constructed the first *publicly* verifiable NIZK scheme for NP from lattice assumptions, although with a limitation to *designated provers*. In another recent work, Quach et al. introduced Laconic Function Evaluation ([55]), which allows functionality qualitatively similar to our high level goal if Bob is *semi-honest*. In contrast, their primitive requires Alice's effort to scale directly with the function description, and the secret decoding information is the knowledge of the function itself.

1.1 Thesis Outline

In Chapter 2, we describe preliminaries on notation, a classic definition of pseudorandom functions, and extended definitions which allow delegation and constraining functionality.

In Chapter 3, we introduce a new cryptographic primitive, a Privately Constrained Testable Pseudorandom Function and define a candidate security model.

In Chapter 4, we describe non-interactive zero-knowledge proof systems, and show how to build one from a Privately Constrained Testable Pseudorandom Function and existing primitives.

In Chapter 5, we review a minimal set of preliminaries on lattices and lattice-based cryptography.

In Chapter 6, we construct the primitive from lattice assumptions, and prove it satisfies the correctness definitions we have proposed. We then show it is secure in an alternate, weaker security model than the model introduced in Chapter 3.

In Chapter 7, we explore some of the proof challenges encountered in the construction of Chapter 6, discuss additional intuitions, and discuss some background details regarding construction and testability of other lattice-based Privately-Constrained PRFs.

In Chapter 8, we summarize the results of the thesis and present open problems left for future work.

Chapter 2

Pseudorandomness and Background

2.1 Notation and Cryptographic Background

To begin, we define some conventions and terminology that will be used throughout the document.

Sets and Probability Distributions. Let $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ denote the sets of the natural numbers, integers, and real numbers. For any $n \in \mathbb{N}$, we define the set $S = [n]$ to be the closed set of values $\{1, \ldots, n\}$. For any set (or vector), we say that S_i is the i-th element or component of S with respect to some ordering. We will also denote uniform distributions over sets, denoted $U(S)$, where we assign a probability measure equal to $1/|S|$ to each $s \in S$.

We denote sampling elements according to a distribution as $s \leftarrow S$. We may also write $s \xleftarrow{\$} S$, which means to sample s uniformly random among all elements of S. We will also refer to B-bounded elements, as any value (or vector) whose support is in the range $[-B, B]$. A B-bounded *distribution* is one such that according to some metric, the largest element belonging to the distribution is B-bounded.

Finally, we define the *statistical distance* between any two distributions X, Y over a countable domain D as $\Delta(X, Y) = \frac{1}{2} \sum_{d \in D} |X(d) - Y(d)|$.

Complexity. We will use standard asymptotic notation to characterize the rate of growth of functions. Additionally we will use $\mathsf{poly}(\cdot)$ to denote some arbitrary deterministic func-

tion $f = O(n^c)$ for a fixed constant c, and PPT to mean algorithms which run in *probabilistic* polynomial time. Any *efficient* algorithm is taken to be bounded by a polynomial. In addition to standard complexity classes $\mathsf{P}, \mathsf{NP}, \mathsf{co{-}NP}$ we will refer to the class NC^1 by the simplified characterization as the set of all polynomial-depth fan-in 2, fan-out 1 binary circuits.

We will often implicitly use polynomial time Turing reductions between algorithms. If A is reducible to B, we write $A \leq B$ to mean that there exists a polynomial time algorithm which solves A given $O(1)$ access to an algorithm which solves B. We also write $A \leq_Q B$ if the reduction is a quantum algorithm. All logarithms are implicitly written in base-2, except where noted.

Cryptography. Let λ denote a security parameter. A *negligible* function $\mathsf{negl}(\lambda)$ with respect to an input λ is bounded from above as follows: $o(\lambda^{-c})$ (e.g. for every constant c). A probability is *overwhelming* if it exceeds $1 - \mathsf{negl}(\lambda)$. Two distributions (X, Y) are *statistically close* if $\Delta(X, Y) < \mathsf{negl}(\lambda)$, and *computationally close* if for every PPT algorithm $\mathcal{A}$, the output distributions of the algorithm are close for both distributions over a bounded quantity of elements in the domain. That is, $|\Pr[\mathcal{A}(1^\lambda, X_\lambda)] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda)]| \leq \mathsf{negl}(\lambda)$. We abbreviate these distances as $\approx_s$ and $\approx_c$ respectively. We say a problem P is *computationally infeasible* if there is no known PPT algorithm which solves P. Inputs to algorithms are sometimes written in unary so that algorithms do not depend logarithmically on the lengths of their *bit representations*.

We will model security as a series of experiments between an adversary $\mathcal{A}$ and a challenger, both represented as PPT algorithms which operate on probability distributions. The *view* of an adversary consists of any information known to them, and any possibly hidden values on which that information depends. In experiments we may restrict the admissibility of adversaries by limiting their capabilities. We say the *advantage* of the adversary is the probability that they can accomplish the *adversarial goal* defined in some experiment.

Experiments are modified through a series of *hybrid arguments* where some component of a distribution is modified, and an argument is made that the change is statistically close or computationally close to the distribution in a preceding variant or "hybrid" of the experiment. If the adversary is able to detect a distributional change which is computationally indistinguishable, we form a reduction which uses the adversary as an oracle to solve the problem in the computational assumption. Otherwise, we make some argument regarding the change of the advantage.

Eventually, we consider a hybrid setting where $\mathcal{A}$ accomplishes the goal ("wins") with some probability close to an "ideal" minimum success probability. The triangle inequality

is invoked to argue that the resulting changes imply $\mathcal{A}$'s advantage must also be negligible in the security parameter in the "real" setting. We may instead also work towards contradiction, and reduce $\mathcal{A}$'s advantage to negligible to argue that $\mathcal{A}$ can be used as a distinguisher between some set of "close" distributions.

We distinguish between *selective* security, where some aspect of the adversarial goal must be fixed and announced by $\mathcal{A}$ at the beginning of an experiment, and *adaptive* security where such restrictions are not present and $\mathcal{A}$ can potentially adapt their attack strategy based on seeing the output of multiple interactions with the challenger. We will consider experiments in two security models. In the *indistinguishability* model, an adversary must distinguish between two possible honestly generated distributions. In the simulation model the adversary must distinguish whether they are interacting with the challenger, or some idealized simulator $\mathcal{S}$ which produces a distribution which is designed to be computationally close to the adversary's view. In the latter case, the distribution is produced given any auxiliary information which is *leaked* to the adversary, but *independently* of their actual inputs.

Finally, we say a function is one-way if it is computationally infeasible to invert.

2.2 Pseudorandom Functions

Pseudorandom functions (PRFs) were first defined in [37]. PRF are functions which should be indistinguishable from a truly random function by any computationally bounded adversary who is allowed to query the evaluation of the function on any polynomial number of inputs (e.g. oracle access).

Definition 2.2.1 (Pseudorandom Function (PRF))**.** *Let $\mathcal{K}$ be a key-space, $\mathcal{X}$ denote an input domain, and $\mathcal{Y}$ denote a range space. A deterministic efficient function family $\mathcal{F} = \{\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ is pseudorandom if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\left| \Pr_{K \leftarrow \mathcal{K}}[\mathcal{A}^{\mathsf{F}(K,\cdot)}(1^\lambda) = 1] - \Pr_{f \xleftarrow{\$} \widehat{\mathcal{F}}}[\mathcal{A}^{f(\cdot)}(1^\lambda) = 1] \right| \leq \mathsf{negl}(\lambda) \tag{2.2.1}$$

for all sufficiently large $\lambda \in \mathbb{N}$, and $\widehat{\mathcal{F}} = \{f : \mathcal{X} \to \mathcal{Y}\}$ the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

In the definition above, we can take $\mathcal{X} = \{0,1\}^n, \mathcal{Y} = \{0,1\}^m$ for $n \leq m$ for example. If $|\mathcal{X}| < |\mathcal{Y}|$ we may distinguish between the *range* $\mathcal{R} \subseteq \mathcal{Y}$ of a PRF F, and its codomain $\mathcal{Y}$.

Then, it also follows that

$$\Pr_{y \xleftarrow{\$} \mathcal{Y}}[y \in \mathcal{R}] \leq |\mathcal{X}|/|\mathcal{Y}|.$$

The security definition above strictly requires that the adversary not possess the key K, as otherwise they have trivial distinguishing power. In richer applications, we would like to delegate the key K to some evaluator $\mathcal{A}$, so that they may evaluate the function themselves — but without trivially losing the security property above. A natural idea is to delegate a limited key which is unable to evaluate the function on certain inputs. Then, we may argue that the same security definition still holds for any points which $\mathcal{A}$ can not evaluate themselves. For example, the PRF described in [37] can easily be made delegateable for prefix-constraints. In particular, for some $x' \in \{0,1\}^\ell$, a key $K[x']$ allows $\mathcal{A}$ to evaluate at all inputs $x \in (x' \,\|\, \{0,1\}^{n-\ell})$. We may also wish to increase granularity and constrain some polynomial number of points which do not share a common prefix. Such a tool is called a *puncturable* PRF, which is modeled by a more complex experiment.

Definition 2.2.2 (Puncturable Pseudorandom Function)**.** *A PRF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is also a puncturable pseudorandom function if it admits an additional key space $\mathcal{K}_P$, and a tuple of efficient algorithms* $(\mathsf{F.Setup}, \mathsf{F.Eval}, \mathsf{F.Puncture})$ *with the following properties:*

$\mathsf{F.Setup}(1^\lambda)$**:** *is a* PPT *algorithm which takes as input a security parameter λ, and outputs a key-space $\mathcal{K}$, a punctured key-space $\mathcal{K}_P$, and a description of* F*.*

$\mathsf{F.Puncture}(K, S)$**:** *is a* PPT *algorithm that takes as input a PRF key $K \in \mathcal{K}$, and a polynomially sized set $S \subset \mathcal{X}$, and outputs a key $K[S] \in \mathcal{K}_P$.*

$\mathsf{F.Eval}(K[S], x)$**:** *is a deterministic efficient algorithm that takes as input a punctured PRF key $K[S] \in \mathcal{K}_P$, an input $x \in \mathcal{X}$, and outputs $y = \mathsf{F}(K, x)$ if $x \notin S$, or it outputs $\perp$*[1] *if $x \in S$.*

Correctness. *A puncturable PRF satisfies correctness if for all $x \in \mathcal{X}$, all polynomially sized sets $S \subset \mathcal{X}$, and all pairs of keys $(K, K[S]) \in \mathcal{K} \times \mathcal{K}_p$, the following holds with probability $1 - \mathsf{negl}(\lambda)$:*

$$\mathsf{F.Eval}(K[S], x) = \begin{cases} \mathsf{F}(K, x) & \text{if } x \notin S \\ \perp & \text{otherwise.} \end{cases}$$

[1] We may represent $\perp$ as some distinct $y' \in \mathcal{Y}$ such that $\mathsf{F}(K, x) \neq y'$ for any $K \in \mathcal{K}, x \in \mathcal{X}$.

Security. *Security for the primitive will hold if the scheme satisfies residual pseudorandomness on the punctured points, meaning for all $x \in S$ the output of* F *is indistinguishable from a uniform function $f : \mathcal{X} \to \mathcal{Y}$, given $K[S]$. We define security as an experiment between a challenger which generates the punctured key, and a* PPT *adversary $\mathcal{A}$ consisting of the following sequence of interactions:*

1. *The challenger chooses $K \xleftarrow{\$} \mathcal{K}$, and a bit $b \xleftarrow{\$} \{0,1\}$.*
2. *$\mathcal{A}$ issues any polynomial number of evaluation queries $\{x_1, \ldots, x_\ell\}$ to an oracle $\mathcal{O}^{\mathsf{F}(K,\cdot)}$.*
3. *$\mathcal{A}$ chooses a challenge set $S^* \subset \mathcal{X}$ with $|S^*| = \mathsf{poly}(\lambda)$, and sends it to the challenger. The challenger computes $K[S^*] = \mathsf{F.Puncture}(K, S^*)$, and outputs $(K[S^*], \{\mathsf{F}(K, x_i^*)\}_{x_i^* \in S})$ if $b = 0$, otherwise if $b = 1$ it outputs $(K[S^*], \{U\} \xleftarrow{\$} \mathcal{Y}^{|S^*|})$.*
4. *$\mathcal{A}$ outputs a guess b', which is the output of the experiment.*

We restrict security to admissible adversaries, satisfying the condition that no element of S^* is ever queried to the evaluation oracle. Then, we say the scheme is a secure puncturable PRF if for all PPT *admissible adversaries $\mathcal{A}$,*

$$\Pr[b' = b] = 1/2 + \mathsf{negl}(\lambda).$$

Of course, the set of punctured items S can consist of a single challenge point x^*. We can generalize this concept completely by defining keys which are constrained by *arbitrary* functions. That is, $K[C]$ (a key constrained with respect to a constraint C) can be used to evaluate F all inputs x for which $C(x) = 1$, but remains pseudorandom on all other inputs. Clearly, this capability subsumes both puncturing and prefix-constraints by taking C to be either an equality or prefix-equality testing function. This notion was independently developed in [18], [46], and [19]. The first instantiation of a circuit-constrained PRF from standard assumptions was due to [25].

Constrained PRFs allow for the construction of primitives such as Broadcast Encryption [18], and are also useful as part of proof techniques [57]. Boneh et al. also formalized *privately constrained* (or *constraint-hiding*) PRFs in [17], which possess the additional property that a constrained key $K[C]$ also hides the constraint function C from any PPT adversary. (We can define privately puncturable PRFs similarly). For circuit-constrained PRFs, we specify the input space $\mathcal{X}$ to be the same as the input space of the constraint C: $\mathcal{X} = \{0,1\}^{\ell_{in}}$.

Definition 2.2.3 (Privately Constrained PRF)**.** *A privately constrained PRF family* $\mathcal{F} = \{\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ *for a constraint class* $\mathcal{C}_\lambda = \{\{0,1\}^{\omega(\lambda)} \to \{0,1\}\}$ *is defined by a tuple of algorithms (*KeyGen*,* Eval*,* $\mathsf{Constrain}$*,* $\mathsf{ConstrainEval}$*):*

$\mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$ *is a* PPT *algorithm that takes as input the security parameter* λ*, a circuit description maximum length* ℓ_C*, an input length* $\ell_{in} = \omega(\lambda)$*, and outputs a secret key parameter* sk *and public parameters* pp*.*

$\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x)$ *is a deterministic algorithm that takes as input a PRF secret key* sk*, an input string* $x \in \{0,1\}^{\ell_{in}}$*, and outputs* $y \in \mathcal{Y}$*.*

$\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$ *is a deterministic algorithm that takes as input a PRF secret key* sk*, a circuit* $C : \{0,1\}^{\ell_{in}} \to \{0,1\}$ *of length at most* ℓ_C*, and outputs a constrained key* $\mathsf{sk}[C]$*.*

$\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x)$ *is a* PPT *algorithm that takes as input a constrained key* $\mathsf{sk}[C]$ *and a string* $x \in \{0,1\}^{\ell_{in}}$*, and outputs* $y \in \mathcal{Y}$*.*

Correctness. *A Constrained PRF* P *preserves functionality for constrained inputs if, for any circuit* $C \in \mathcal{C}_\lambda$*,* $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$*, all inputs* $x \in \{0,1\}^{\ell_{in}}$ *for which* $C(x) = 1$*, we have*

$$\Pr[\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C), x) = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x)] > 1 - \mathsf{negl}(\lambda).$$

Security. *We restrict the definitions below to the case of a single constrained key query to avoid requiring stronger assumptions (cf. Remark 2.2.1).*

1. *Residual Pseudorandomness. Given access to three oracles (constrained key, evaluation and challenge), the output of* $\mathsf{Eval}_{\mathsf{pp}}(\cdot)$ *is indistinguishable from random for any inputs not queried to the evaluation oracle, or is producible by* $\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], \cdot)$*.*
   - *Constrained key oracle. Given a circuit* $C \in \mathcal{C}_\lambda$*, the challenger outputs a constrained key* $\mathsf{sk}[C] \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$*.*
   - *Evaluation Oracle. Given an input* $x \in \{0,1\}^{\ell_{in}}$ *outputs* $y \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x)$*.*
   - *Challenge Oracle. Given an input* $x^* \in \{0,1\}^{\ell_{in}}$*, the challenger samples* $b \xleftarrow{\$} \{0,1\}$*, and outputs* $y \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x^*)$ *if* $b = 1$*, and* $y \xleftarrow{\$} \mathcal{Y}$ *if* $b = 0$*.*

The adversary begins by querying for a constrained key, and is then allowed a polynomial amount of evaluation queries. $\mathcal{A}$ then queries the Challenge Oracle on a selected input x^*, and receives y. At the end of the experiment, $\mathcal{A}$ outputs b', and wins if $b' = b$. An adversary $\mathcal{A}$ is admissible if x^* is never queried to the evaluation oracle, and $C(x^*) = 0$, to prevent trivial distinguishability. Residual pseudorandomness is satisfied if the winning probability of any PPT *adversary is bounded by* $1/2 + \mathsf{negl}(\lambda)$.

2. *Indistinguishability-based constraint privacy. Given access to two oracles (constrained key and evaluation), for any selectively chosen $C_0, C_1 \in \mathcal{C}_\lambda$, the constrained keys $\mathsf{sk}[C_0]$, $\mathsf{sk}[C_1]$ are computationally indistinguishable.*

   - *Constrained key oracle. Given a pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, the challenger samples a bit $b \xleftarrow{\$} \{0,1\}$ and outputs a constrained key $\mathsf{sk}[C_b] \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C_b)$.*
   - *Evaluation Oracle. Given an input $x \in \{0,1\}^{\ell_{in}}$, and a bit b, outputs $y \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x)$.*

   The adversary begins by querying the challenger with two circuits $C_0, C_1 \in \mathcal{C}_\lambda$, and receives back $\mathsf{sk}[C_b]$ for a random bit b. $\mathcal{A}$ is then allowed a polynomial amount of evaluation queries. At the end of the experiment, $\mathcal{A}$ outputs b', and wins if $b' = b$. For admissibility we require that $C_0(x) = C_1(x)$ for all inputs x queried to the evaluation oracle, to prevent trivial distinguishability. Constraint privacy is satisfied if the winning probability of any PPT *adversary is bounded by* $1/2 + \mathsf{negl}(\lambda)$.

Remark 2.2.1 (Security (Constraint Privacy)). Canetti and Chen [27] showed that a scheme which satisfied security for multiple distinct constraint queries would imply the much stronger primitive of Indistinguishability Obfuscation (iO) [11]. To separate the analysis from that of iO, we limit admissibility to a single constraint query (which are always selectively chosen).

Canetti and Chen also showed for selectively chosen constraint, the above indistinguishability-based constraint privacy and pseudorandomness definitions are equivalent to the following simulation based definition. In particular, simulation implies indistinguishability by a straightforward hybrid argument where C_0 is switched to C_1 by being simulated in an intermediate step.

Definition 2.2.4 (Privately Constrained PRF Simulation Security)**.** *For any stateful* PPT *algorithm* $\mathcal{A}$*, there exists a* PPT *stateful simulator* $\mathcal{S}$ *such that*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathcal{S}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

$\mathcal{A}$ may ask a single constraint query for some circuit $C \in \mathcal{C}_\lambda$ followed by a polynomially bounded number of evaluation queries. In the ideal experiment, the constrained key is produced by a simulator which receives only the size of the constraint ℓ_C. In the ideal experiment, the simulator also learns an indicator bit $d_x = C(x)$ to enforce consistency.[2] *If $d_x = 0$, $\mathcal{S}$ answers evaluation queries by (statefully) sampling a uniformly random value from the co-domain. Otherwise, it answers by evaluating using the simulated key. The output of the experiment is the output bit of $\mathcal{A}$.*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)}$:

1: $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$
2: $\mathcal{A} \rightarrow C$
3: $\mathsf{sk}[C] = \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$
4: $\mathcal{A} \leftarrow \mathsf{sk}[C]$
5: *Repeat:*
6: $\quad \mathcal{A} \rightarrow x$
7: $\quad y = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x)$
8: $\quad \mathcal{A} \leftarrow y$
9: $\mathcal{A} \rightarrow b$; *Output* b

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathcal{S}}(1^\lambda)}$:

1: $\mathcal{S} \leftarrow 1^\lambda$
2: $\mathcal{A} \rightarrow C$
3: $\mathsf{sk}[\widetilde{C}] = \mathcal{S}(1^{\ell_C})$
4: $\mathcal{A} \leftarrow \mathsf{sk}[\widetilde{C}]$
5: *Repeat:*
6: $\quad \mathcal{A} \rightarrow x$; $d_x = C(x)$
7: $\quad$ ***if*** $d_x = 1$ ***then***
8: $\quad\quad y' = \mathcal{S}(x, d_x)$
9: $\quad$ ***else***
10: $\quad\quad y' \xleftarrow{\$} \mathcal{Y}$
11: $\quad$ ***end if***
12: $\quad \mathcal{A} \leftarrow y'$
13: $\mathcal{A} \rightarrow b$; *Output* b

Recently, concrete instantiations based on LWE (Definition 5.2.4) have been achieved in [27], [22], [54], [31].

[2] We can consider an alternate formulation where the adversary provides the indicator bit, and the experiment aborts in the event of an inconsistency. However, this is equivalent to restricting the admissibility of the adversary and having the simulator learn b_x from the environment.

Chapter 3

Range-testable Pseudorandom Functions

3.1 Definitions

Here, we formally define the notion of a Privately Constrained Testable PRF (PCT-PRF), and consider a relevant security model.

Definition 3.1.1 (Privately Constrained Testable PRF)**.** *A privately constrained testable PRF family* $\mathcal{P} = \{\mathsf{P} : \mathcal{K} \times \mathcal{X} \times \{0,1\} \to \mathcal{R}_0 \cup \mathcal{R}_1\}$ *for a constraint class* $\mathcal{C}_\lambda = \{\{0,1\}^{\omega(\lambda)} \to \{0,1\}\}$ *is defined by a tuple of algorithms (*KeyGen*,* Eval*,* $\mathsf{Constrain}$*,* $\mathsf{ConstrainEval}$*,* Test*). With respect to* Eval *as defined below, we have* $\mathcal{R}_b = \cup_{x \in \mathcal{X}} \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, b, x)$ *and* $\mathcal{R}_0 \cup \mathcal{R}_1 \subseteq \mathcal{Y}$*:*

$\mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$ *is a* PPT *algorithm that takes as input the security parameter* λ*, a circuit description maximum length* ℓ_C*, an input length* ℓ_{in}*, and outputs a secret key parameter* sk *and public parameters* pp*.*

$\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b)$ *is an efficient deterministic algorithm that takes as input a PRF secret key* sk*, an input string* $x \in \{0,1\}^{\ell_{in}}$*, a bit* $b \in \{0,1\}$*, and outputs* $y \in \mathcal{R}_b$*.*

$\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$ *is a* PPT *algorithm that takes as input a PRF secret key* sk*, a circuit* $C : \{0,1\}^{\ell_{in}} \to \{0,1\}$ *of length at most* ℓ_C*, and outputs a constrained key* $\mathsf{sk}[C]$*.*

$\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x)$ *is an efficient deterministic algorithm that takes as input a constrained key* $\mathsf{sk}[C]$ *and a string* $x \in \{0,1\}^{\ell_{in}}$*, and outputs* $y \in \mathcal{R}_b$ *for* $b = C(x)$*.*

$\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y)$ *is an efficient deterministic algorithm that takes as input a secret key* sk*, a bit* b*, and a value* $y \in \mathcal{Y}$ *and outputs a single bit.*

Correctness. *Fix any* $C \in \mathcal{C}_\lambda$*. A constrained PRF* P *has evaluation correctness if for all* $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$*,* $\mathsf{sk}[C] \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$ *all inputs* $x \in \{0,1\}^{\ell_{in}}$*,*

$$\Pr_{\mathsf{sk},x,b}[\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x) = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, C(x))] > 1 - \mathsf{negl}(\lambda).$$

We define Completeness and Unique Testability properties next. For completeness, we require that for any $y \in \mathcal{R}_b$, the algorithm $\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b', y)$ outputs 1 if $b = b'$.

Completeness. A testable PRF satisfies testing completeness if for all $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$, $x \in \{0,1\}^{\ell_{in}}$, and $b \in \{0,1\}$,

$$\Pr_{\mathsf{sk},x,b}[\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b)) = 1] > 1 - \mathsf{negl}(\lambda).$$

Next, we define a unique testability over the co-domain $\mathcal{Y}$, and hence both ranges of the PRF ($\mathcal{R}_0 \cup \mathcal{R}_1$), in that any output is unambiguously testable as belonging to a single range $\mathcal{R}_b$ with overwhelming probability.

Unique Testability. A testable PRF satisfies unique testability if

$$\Pr_{\mathsf{sk},y\in\mathcal{Y}}[\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, 0, y) = \mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, 1, y) = 1] < \mathsf{negl}(\lambda).$$

Security. We restrict the definitions below to the case of a single constrained key query to avoid requiring stronger assumptions (cf. Remark 2.2.1).

1. Residual Pseudorandomness. Given access to four oracles (constrained key, evaluation, testing and challenge), the output of $\mathsf{Eval}_{\mathsf{pp}}(\cdot)$ is indistinguishable from random for any inputs not queried to the evaluation oracle, or not producible by $\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], \cdot)$.

    - Constrained key oracle. Given a circuit $C \in \mathcal{C}_\lambda$, the challenger outputs a constrained key $\mathsf{sk}[C] \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$.

- Evaluation Oracle. Given an input $x \in \{0,1\}^{\ell_{in}}$, and a bit b, outputs $y \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b)$.
- Testing Oracle. Given a $y \in \mathcal{Y}$, and a bit b, outputs $\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y)$.
- Challenge Oracle. Given an input $x^* \in \{0,1\}^{\ell_{in}}$, the challenger sets $b = C(x^*)$ and samples $c \xleftarrow{\$} \{0,1\}$. Finally it outputs $y \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x^*, 1-b)$ if $c = 1$, and $y \xleftarrow{\$} \mathcal{Y}$ if $c = 0$.

The adversary begins by querying for a constrained key, and is then allowed a polynomial amount of evaluation and testing queries. $\mathcal{A}$ then queries the Challenge Oracle on a selected input x^*, and receives y. At the end of the experiment, $\mathcal{A}$ outputs c', and wins if $c' = c$. An adversary $\mathcal{A}$ is admissible if $(x^*, 1 - C(x^*))$ is never queried to the evaluation oracle. Residual Pseudorandomness is satisfied if the winning probability of any PPT adversary is bounded by $1/2 + \mathsf{negl}(\lambda)$.

2. Indistinguishability-based constraint privacy. Given access to three oracles (constrained key, evaluation and testing oracles), for any selectively chosen $C_0, C_1 \in \mathcal{C}$, the constrained keys $\mathsf{sk}[C_0]$, $\mathsf{sk}[C_1]$ are computationally indistinguishable.

   - Constrained key oracle. Given a pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, the challenger samples a bit $d \xleftarrow{\$} \{0,1\}$ and outputs a constrained key $\mathsf{sk}[C_d] \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C_d)$.
   - Evaluation Oracle. Given an input $x \in \{0,1\}^{\ell_{in}}$, and a bit b outputs $y \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b)$.
   - Testing Oracle. Given a $y \in \mathcal{Y}$, and a bit b, outputs $\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y)$.

   The adversary begins by querying the challenger with two circuits C_0, C_1, and receives back $\mathsf{sk}[C_d]$ for a random bit d. $\mathcal{A}$ is then allowed a polynomial amount of evaluation and testing queries. At the end of the experiment, $\mathcal{A}$ outputs d', and wins if $d' = d$. For admissibility we require that $C_0(x) = C_1(x)$ for all inputs x, to prevent trivial distinguishability. Constraint privacy is satisfied if the winning probability of any PPT adversary is bounded by $1/2 + \mathsf{negl}(\lambda)$.

3. Conditional One-wayness. We say the PCT-PRF is one-way conditioned on the secret key if for any PPT adversary $\mathcal{A}$, all $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$, $x \in \{0,1\}^{\ell_{in}}$, $b \in \{0,1\}$, we have

$$\Pr_{\mathsf{sk},x,b}[\mathcal{A}(\mathsf{sk}, \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b)) \to x] = \mathsf{negl}(\lambda).$$

Ignoring the testability properties, the definition of the primitive is quite similar to Definition 2.2.3, except that the residual pseudorandomness property is "two-sided". The PRF definition above has two independent ranges corresponding to $b = 0$ and $b = 1$ which are mapped to independent of any constraint. Recall that in a standard constrained PRF, the range of the function under sk and $\mathsf{sk}[C]$ intersect for all x such that $C(x) = 1$, where the range under $\mathsf{sk}[C]$ may be *induced* by the constraining algorithm. Instead, the primitive above has two ranges which are well defined and testable *independently* of any constraint (which may not be guaranteed in a regular PC-PRF) — and the range under some constrained key $\mathsf{sk}[C]$ may intersect both of them.

In exposition, we will refer to R_b the "b-Range" of the PRF. Hence, $\mathsf{Range}(\mathcal{P}) = (\mathcal{R}_0 \cup \mathcal{R}_1) \subseteq \mathcal{Y}$. Note that the unique testability property implies $|\mathcal{R}_0 \cap \mathcal{R}_1| = \mathsf{negl}(\lambda)$, and therefore that $|\mathcal{X}|/|\mathcal{Y}| \leq 1/2$. Rather than having "constrained" and "unconstrained" outputs as in standard constrained PRFs, and requiring $\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x)$ to be indistinguishable from random for all inputs x with $C(x) = 0$ given $\mathsf{sk}[C]$ — we require the output of the function to be indistinguishable from random for all values not producible using the constrained key. That is, for any input x with $C(x) = b$, $\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, 1-b)$ should *also* be indistinguishable from random.

Also, note that the circuit privacy property has the stronger requirement that the constraints queried be functionally equivalent, rather than have the same outputs for all values queried to the evaluation oracle — since the testing oracle can be used on outputs not returned from evaluation queries and generated using the constrained key. This trivially allows for distinguishing between any constraints with differing functionality. Given the extra requirements of functional equivalence, the testing oracle does not provide any additional capability not already given by the evaluation oracle.

The inclusion of a testing oracle in this definition is done in order to preserve correctness during hybrid experiments for applications of the primitive where we may wish to switch between functionally equivalent constraints. Otherwise simulation arguments cannot go through directly since an adversary gains trivial distinguishing capability.

Remark 3.1.1 (Usefulness of Functional Equivalence). The additional restriction of functional equivalence is only introduced to prevent trivial distinguishing capabilities provided by the testing oracle (which we will rely on for the proof of security in Construction 4.1.1). At first glance, it may appear useless if to only support indistinguishability for circuits which have identical input and output distributions. However, counterintuitively we can exploit a subtle observation that we can use circuits whose functionality differs on inputs *outside* of the input distribution. This technique follows from the techniques introduced by Sahai and Waters [57], known as *punctured programs*.

Chapter 4

Applications

4.1 Designated-verifier Non-interactive Zero-knowledge

In this section we will show how to construct a designated-verifier non-interactive zero-knowledge (NIZK) proof system where any protocol participant is able to use a shared public parameter to produce *zero-knowledge* proofs. Informally, these are proofs which irrefutably attest to the truth of certain mathematical statements without revealing any significant information about the *why* the statement is true.

A *designated-verifier* scheme is a special case of a *privately verifiable* proof system in which verification capability is restricted. When the verifier is *designated* we require that the setup (preprocessing step) of the scheme (which produces the public parameters) is done in a trustworthy manner (or by a trusted-third party). This is crucial for zero-knowledge, as we can envision a malicious verifier which produces pathologically malformed public parameters which cause any proofs output by the scheme to trivially leak information, or allow for the proving of false statements. This weakness is commonly called *trusted-setup*, and is also a challenge when designing *publicly* verifiable proof systems[3].

We can now formally define, and construct a Designated-verifier NIZK for NP *languages* $\mathcal{L}$ (mathematical problems), which are defined together with a relation $R_{\mathcal{L}} \to \{0, 1\}$. We often omit the subscript on $R_{\mathcal{L}}$ when it is obvious from context. A *statement* (a problem instance) x belongs to $\mathcal{L}$ if and only if there exists a *witness* (or "certificate") w such that $R(x, w) = 1$. That is, $x \in \mathcal{L} \iff \exists w : R(x, w) = 1$. We say that the language $\mathcal{L}$ is in NP

[3]Non-interactive proof systems which allow preprocessing, are publicly provable, *and* publicly verifiable capture the CRS ("Common Reference String") model

if for any statement $x \in \mathcal{L}$, there exists a witness w with $|w| = \mathsf{poly}(|x|)$ such that $R(x, w)$ can be evaluated in time $\mathsf{poly}(|x|, |w|)$. Our construction will resemble a decomposed version of the NIZK construction in [57].

Definition 4.1.1 (Designated-verifier NIZK Argument)**.** *Let R be an* NP *relation for a language $\mathcal{L}$. A non-interactive zero-knowledge (NIZK) argument for $\mathcal{L}$ in the Designated-verifier model is a computationally sound proof system consisting of the following three algorithms:*

- $\mathsf{N.Setup}(1^\lambda, R) \to (pk, vk)$*: is a* PPT *algorithm. Given as input the security parameter λ and the* NP *relation R, the setup algorithm outputs a proving/verification key pair (pk, vk).*
- $\mathsf{N.Prove}(pk, x, w) \to \pi$*: is a* PPT *algorithm. Given as input the proving key pk, a statement x and witness w, the proving algorithm outputs a proof π.*
- $\mathsf{N.Verify}(vk, x, \pi) \to \{0, 1\}$*: is a deterministic efficient algorithm. Given as input the verification key vk, a statement x and proof π, the verification algorithm outputs 1 if and only if the proof is accepted.*

The scheme N *must also satisfy the following properties:*

Completeness. *For all (x, w) for which $R(x, w) = 1$, and any $(pk, vk) \leftarrow \mathsf{N.Setup}(1^\lambda, R)$:*

$$\Pr_{(x,w)}[\mathsf{N.Verify}(vk, x, \pi) = 1 \mid \pi \leftarrow \mathsf{N.Prove}(pk, x, w)] = 1 - \mathsf{negl}(\lambda).$$

Soundness. *For any* PPT *bounded adversary $\mathcal{A}$, for any $(pk, vk) \leftarrow \mathsf{N.Setup}(1^\lambda, R)$, and for all statements x:*

$$\Pr[x \notin \mathcal{L} \wedge \mathsf{N.Verify}(vk, x, \pi) = 1 \mid (x, \pi) \leftarrow \mathcal{A}^{\mathsf{N.Verify}(vk,\cdot)}(pk)] = \mathsf{negl}(\lambda).$$

Zero-knowledge. *For any* PPT *bounded adversary $\mathcal{A}$, there exists a* PPT *simulator $\mathcal{S}$ such that for $(pk, vk) \leftarrow \mathsf{N.Setup}(1^\lambda, R)$, and all (x, w) with $R(x, w) = 1$,*

$$\mathcal{A}(\pi \leftarrow \mathsf{N.Prove}(pk, x, w)) \approx_c \mathcal{A}(\pi \leftarrow \mathcal{S}(pk, vk, x)).$$

Construction 4.1.1 (Designated-verifier NIZK Argument from PCT-PRF)**.**

- *Let* λ *be a security parameter*
- *Let* $R : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$*, for* $n = |x|, m = |w|$)
- *Let* $\mathsf{P} = (\mathsf{P.KeyGen}, \mathsf{P.Eval}, \mathsf{P.Constrain}, \mathsf{P.ConstrainEval}, \mathsf{P.Test})$ *be a PCT-PRF with input size* $\ell_{in} = n + m + \lambda$*.*
- *Let* $\mathsf{F} = (\mathsf{F.KeyGen}, \mathsf{F.Puncture}, \mathsf{F.Eval})$ *be a puncturable PRF family with input size* $n + 1$ *and output size* $\psi = \mathsf{poly}(\lambda, n)$
- *Let* C *be the circuit in Fig. 4.1*

For convenience, we define the following abbreviated notation. Let $\mathsf{Test}(i, \pi_i)$ *be an algorithm which outputs* b *if* $\mathsf{P.Test}_{\mathsf{pp}_i}(\mathsf{sk}_i, b, \pi_i)$ *outputs* 1 *for exactly one* $b \in \{0,1\}$*, and* $\perp$ *otherwise. Additionally, we take* $\mathsf{F}(K, x)$ *to denote* $\mathsf{F.Eval}(K, x)$*.*

Construction Overview. The starting intuition behind the construction is to use the witness relation R as the constraint circuit C. Then, any PRF input $(x \parallel w)$ will produce a value $y \in \mathcal{R}_1$. The designated-verifier can then use sk to test whether y is in fact in the correct range. This approach only supports a single use. So, we instead consider a a hypothetical multi-bit output constraint which outputs elements in a range indexed by a *puncturable* PRF $\mathsf{F}(K, (R(x, w) \parallel x))$. In this case a valid proof would land in a fixed pseudorandom range which depends on the statement x, and whether $x \in \mathcal{L}$. Intuitively, forging a proof in this setting implies predicting the PRF output for $\mathsf{F}(K, (1 \parallel x))$ for $x \notin \mathcal{L}$, i.e. for which $R(x, w) = 0$ for all w. In the proof we will puncture K on some input $(1 \| x^*)$, which preserves functionality of the circuit perfectly, while remapping the target required for the forgery to something uniformly random.

Of course, the definition of the PCT-PRF is limited to predicates, but we can replicate this intuition by decomposing the constraint into many constrained keys which output single bits of $\mathsf{F}(K, (R(x, w) \parallel x))$. We will use a different sk_i for each component to avoid the issue raised in Remark 2.2.1. We will also append a salt onto the inputs which is ignored by the constraint (implicitly treated as a no-op) to facilitate worst-case zero-knowledge which would be otherwise impossible given the deterministic nature of the PCT-PRF.[4] The formal construction proceeds below.

[4]The salt is also required to satisfy the requirement of the constraint belonging to $\mathcal{C}_\lambda$, as the size of the domain for a language with small statements may have size considerably smaller than the security parameter. In which case, Conditional One-wayness cannot hold in a computational sense.

$\mathsf{N.Setup}(1^\lambda, R) \to (pk, vk)$**:**

1. Let $K \leftarrow \mathsf{F.Setup}(1^\lambda)$.
2. For $i \in [\psi]$, let $(\mathsf{sk}_i, \mathsf{pp}_i) \leftarrow \mathsf{P.KeyGen}(1^\lambda, |R| + |\mathsf{F}|, n + m + \lambda)$.
3. For $i \in [\psi]$, let $\{\mathsf{sk}_i[C_i] \leftarrow \mathsf{P.Constrain}_{\mathsf{pp}_i}(\mathsf{sk}_i, C_i)\}_{i\in[\psi]}$
4. Output $(pk, vk) = (\{\mathsf{sk}_i[C_i]\}_{i\in[\psi]}, (K, \{\mathsf{sk}_i\}_{i\in[\psi]}))$

$\mathsf{N.Prove}(pk, x, w) \to \pi$**:**

1. If $R(x, w) = 0$, output $\perp$
2. Parse $pk = \{\mathsf{sk}_i[C_i]\}_{i\in[\psi]}$
3. For $i \in [\psi]$:
   (a) Sample $r_i \xleftarrow{\$} \{0,1\}^\lambda$
   (b) Let $u_i = (x \parallel w \parallel r_i)$
4. Output $\pi = \{\mathsf{P.ConstrainEval}_{\mathsf{pp}_i}(\mathsf{sk}_i[C_i], u_i)\}_{i\in[\psi]}$

$\mathsf{N.Verify}(vk, x, \pi) \to \{0,1\}$**:**

1. Parse $vk = (K, \{\mathsf{sk}_i\}_{i\in[\psi]})$.
2. Parse $\pi = (\pi_1, \ldots, \pi_\psi)$
3. Evaluate $\pi' = (\mathsf{Test}(1, \pi_1) \parallel \cdots \parallel \mathsf{Test}(\psi, \pi_\psi))$
4. Compute $z = \mathsf{F}(K, (1 \parallel x))$
5. Accept if $\pi' = z$

Prove NIZK

Constants: PRF key K
Input: u_i
1: Parse[5]$(x, w, r) = u_i$
2: Compute $\beta = \mathsf{F}(K, R(x, w) \parallel x)$
Output: Output β_i

Figure 4.1: Constraint circuit C_i

[5]Note that only an arbitrary no-op is applied to r, and its value is not used functionally inside the constraint. The constraint computes the entire evaluation of $\mathsf{F}(K, \cdot)$, but only outputs the i-th bit of its output.

4.1.1 Security

Theorem 4.1.1. *If* P *is a correct PCT-PRF with testing completeness, and* F *is a secure puncturable PRF with respect to Definition 2.2.2, then the NIZK in Construction 4.1.1 satisfies completeness.*

Proof. Completeness follows from the correctness and completeness properties of the PCT-PRF.

Let $z = \mathsf{F}(K, (1 \,\|\, x))$. For any (x, w) such that $R(x, w) = 1$ and $r_i \in \{0,1\}^\lambda$, the output of the predicate $C_i(x \,\|\, w \| \, r_i)$ is the i-th output bit of the internal PRF F: $\beta_i = \mathsf{F}(K, (1 \,\|\, x))_i$ (i.e. z_i). Let $u_i = (x \| \, w \| \, r_i)$ for $i \in [\psi]$. By the correctness property of PCT-PRFs we have that $\pi_i = \mathsf{P.ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C_i], u_i) = \mathsf{P.Eval}_{\mathsf{pp}}(\mathsf{sk}_i, u_i, \beta_i)$ with probability $1 - \mathsf{negl}(\lambda)$. Then, by testing completeness of the PCT-PRF it follows that $\mathsf{P.Test}_{\mathsf{pp}}(\mathsf{sk}, \pi_i, \beta_i) = 1$.

Recall the abbreviated notation from the construction, where $\mathsf{Test}(i, \pi_i)$ outputs b if $\mathsf{P.Test}_{\mathsf{pp}_i}(\mathsf{sk}_i, b, \pi_i)$ outputs 1 for exactly one $b \in \{0,1\}$, and $\perp$ otherwise. Then, the equalities above hold with overwhelmingly high probability for all $i \in [\psi]$.

It follows that $\pi' = (\mathsf{Test}(1, \pi_1) \,\|\, \cdots \,\|\, \mathsf{Test}(\psi, \pi_\psi)) = z$. By definition, the condition for accepting the proof is $\pi' = z$. We conclude that the construction satisfies completeness with the required bound.

□

Theorem 4.1.2. *If* P *is a PCT-PRF with constraint indistinguishability, and* F *is a puncturable PRF with respect to Definition 2.2.2, the NIZK in Construction 4.1.1 is computationally sound against selectively chosen statements.*

Proof. We prove the claim through a series of hybrid arguments. For the remainder of the theorem, x^* is some selectively-chosen challenge statement.

H_0**:** This is the real setting.

$\mathsf{H}_{1,i}$**:** For all $j \leq i$, the constraint circuit C_j instead contains a punctured PRF key $K[1 \| x^*]$.

H_2**:** Steps 3-5 of $\mathsf{N.Verify}$ calculate a uniformly random value z^* and compare $\pi' = z^*$ instead of evaluating F and comparing $\pi' = \mathsf{F}(K, 1 \,\|\, x^*)$.

Lemma 4.1.1. *If* P *is a PCT-PRF scheme with constraint privacy with respect to indistinguishability security of Definition 3.1.1, then for all* $\mathcal{A}$ *we have* $|\Pr[\mathsf{H}_{1,i}(\mathcal{A}) = 1] - \Pr[\mathsf{H}_{1,i-1}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. In the real setting, the verifier receives a puncturable PRF key K from the trusted authority, and the set of secret keys $\{\mathsf{sk}_i\}_{i\in\psi}$, and $p = (\{\mathsf{sk}_i[C_i]\}_{i\in[\psi]}$, the public parameter. The adversary wins if they produce $\{(\pi^*, x^*) \mid \mathsf{N.Verify}(vk, \pi^*, x^*) = 1 \wedge x^* \notin \mathcal{L}\}$. Let $\mathsf{H}_0(\mathcal{A})$ denote the advantage of the adversary $\mathcal{A}$ in attacking the soundness of the scheme on a selectively chosen statement x^* in the real setting.

The only difference between H_0 and $\mathsf{H}_{1,0}$ is that the PRF key K has been punctured on $(1 \,\|\, x^*)$. However, since $x^* \notin \mathcal{L}$, there does not exist w^* such that $R(x^*, w^*) = 1$, and the PRF can never be called on the input $(1 \,\|\, x^*)$ in program C_i. Hence, the behaviour of both constraints is functionally equivalent. Therefore, if there is a difference in advantage between the two hybrids, we can use $\mathcal{A}$ to create an efficient adversary $\mathcal{B}$ against the constraint privacy of the PCT-PRF.

$\mathcal{B}$ samples a key K for the PRF F. $\mathcal{B}$ creates a circuit description of C_i (Fig. 4.1) with K fixed, or with $K[1 \,\|\, x^*]$ fixed (e.g. C_i'), and issues a constraint query to the PCT-PRF challenger with both circuits. After receiving the constrained key for one of $\{C_i, C_i'\}$, $\mathcal{B}$ answers A's verification queries for any submitted (x, π) by issuing testing queries to the PCT-PRF challenger for π_i since it does not have the testing key sk_i. For all other $i' \in [\psi]$, it performs the tests itself.

Consider the case when $i = 1$. If the challenger chooses the original circuit C_i, we are in H_0, otherwise we are in $\mathsf{H}_{1,1}$. Any non-negligible change in advantage for $\mathcal{A}$ implies $\mathcal{B}$ has non-negligible advantage against the constraint privacy of the PCT-PRF. We repeat through a series of hybrid arguments over $i \in [\psi]$ until all ψ circuits are changed and we are in $\mathsf{H}_{1,\psi}$.

□

Lemma 4.1.2. *If* F *is a selectively secure puncturable PRF according to Definition 2.2.2, then for all* $\mathcal{A}$ *we have* $|\Pr[\mathsf{H}_2(\mathcal{A}) = 1] - \Pr[\mathsf{H}_{1,\psi}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. For any $\mathcal{A}$ with a non-negligible change in advantage, we create an efficient adversary $\mathcal{B}$ against the security of the puncturable PRF. $\mathcal{B}$ plays the role of the NIZK verifier. Before creating pk, $\mathcal{B}$ issues a punctured key query (punctured on $(1 \,\|\, x^*)$) to the punctured PRF challenger, and receives back $K[1 \,\|\, x^*]$ and a challenge value z^*. (Recall above that x^* is known to the reduction in the selective security model.) The punctured key is otherwise functional for verifying all other statements $x \neq x^*$.

$\mathcal{B}$ then samples constrained keys $\{\mathsf{sk}_i[C_i']\}_{i\in\psi}$ using $K[1 \,\|\, x^*]$, and provides them to $\mathcal{A}$.

When verifying the proof π^*, $\mathcal{B}$ compares against the value z^* directly. If $\mathcal{A}$ wins, $\mathcal{B}$ infers that it obtained the real PRF output $\mathsf{F}(K, 1 \,\|\, x^*)$ from the challenger and outputs 1. Otherwise, it outputs 0 to indicate z^* was uniformly random.

If the real value was returned, then we are exactly in $\mathsf{H}_{1,\psi}$, otherwise if z^* is uniformly random then we are in H_2. Therefore, if there is a difference in advantage between the two hybrids, we can use $\mathcal{A}$ to create an efficient adversary $\mathcal{B}$ against the security privacy of the puncturable PRF.

□

Finally, through the triangle inequality, Lemma 4.1.1 and Lemma 4.1.2, it follows that for any PPT adversary $\mathcal{A}$ the change in advantage between H_0 and H_2 is at most negligible in the security parameter.

However, in H_2, forging requires producing a uniformly random output, which is exponentially unlikely. Therefore, the scheme satisfies computational soundness.

□

Theorem 4.1.3. *If* P *is a one-way PCT-PRF given* sk*, and satisfies correctness as in Definition 3.1.1, then the NIZK in Construction 4.1.1 is zero-knowledge.*

Proof. We begin by defining the behaviour of evaluation oracles which will be used by the simulator to invoke one-wayness.

$\{\mathcal{O}^*_{i,b}(\cdot)\}$ provides oracle access to the function $\mathsf{P.Eval}_{\mathsf{pp}}(\mathsf{sk}_i, b, \cdot)$ for $i \in [\psi], b \in \{0,1\}$. Additionally, the oracles provide a challenge functionality as follows: when responding to an evaluation query on an input x, it first flips a coin and obtains a value $c \in \{0,1\}$. If $c = 0$ the oracle returns the evaluation on x, otherwise if $c = 1$ it outputs a uniformly random element in $\mathcal{R}_b$.

The behaviour of $\mathcal{S}$ is defined as:

1. Compute $z^* = \mathsf{F}(K, (1 \,\|\, x))$
2. Sample $u^*_i \xleftarrow{\$} \{0,1\}^{(n+m+\lambda)}$
3. Output $(x^*, \pi = \{\mathcal{O}^*_{i,z^*_i}(u^*_i\})$.

Now, we prove the claim through a series of hybrid arguments.

H_0**:** All proof queries are answered by computing $\mathsf{N.Prove}(pk, x, w)$.

$\mathsf{H}_{1,i}$**:** For $i \in [\psi]$ the proof oracle query which produces $\pi_{i'}$ for all $i' \leq i$ is instead produced by evaluating $\mathsf{P.Eval}_{\mathsf{pp}}(\mathsf{sk}_i, z_i, \cdot)$ on an honestly generated input $(x \parallel w \parallel r)$ (without computing $\mathsf{P.ConstrainEval}_{\mathsf{pp}_i}(\mathsf{sk}_i[C_i], \cdot)$ and bypassing the constrained key).

$\mathsf{H}_{2,i}$**:** For $i' \leq i$, the simulator outputs π_i by evaluating $\mathsf{P.Eval}_{\mathsf{pp}}(\mathsf{sk}_i, b, u_i^*)$ on a uniformly random input $u_i^* \in \{0,1\}^{(n+m+\lambda)}$.

H_3**:** The proof is completely generated by $\mathcal{S}$.

Lemma 4.1.3. *If* P *is a PCT-PRF scheme with evaluation correctness, then* $|\Pr[\mathsf{H}_{1,i}(\mathcal{A}) = 1] - \Pr[\mathsf{H}_{1,i-1}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. By correctness of the PCT-PRF, if $R(x, w) = 1$, set $z = \mathsf{F}(K, 1 \parallel x)$, and compute:

$$\begin{aligned} \mathsf{sk}_i[C_i] &= \mathsf{P.Constrain}_{\mathsf{pp}_i}(\mathsf{sk}_i, C_i) \\ y &= \mathsf{P.ConstrainEval}_{\mathsf{pp}_i}(\mathsf{sk}_i[C_i], (u_i))) \\ y' &= \mathsf{P.Eval}_{\mathsf{pp}_i}(\mathsf{sk}_i, u_i), z_i) \end{aligned}$$

Then:

$$\Pr[y \neq y'] < \mathsf{negl}(\lambda)$$

Hence, the change is statistically close.

□

Lemma 4.1.4. *If* P *is a one-way PCT-PRF, then for all* $\mathcal{A}$ *we have* $|\Pr[\mathsf{H}_{2,i}(\mathcal{A}) = 1] - \Pr[\mathsf{H}_{2,i-1}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. First, we see that $z = \mathsf{F}(K, x)$ can be generated independently of the witness for any statement $x \in \mathcal{L}$. Furthermore, the outputs are no longer functionally dependent on the real public parameters pk since $\mathcal{O}^*_{i,b}$ outputs values in the correct range of $R_b(\mathsf{P}_i)$ regardless of the constraint circuits C_i.

Consider the case when $i = 1$. For any adversary $\mathcal{A}$ whose advantage differs between $\mathsf{H}_{1,\psi}$ and $\mathsf{H}_{2,i}$, we create an algorithm $\mathcal{B}$ which breaks the one-wayness of P.

$\mathcal{B}$ computes $z = \mathsf{F}(K, (1 \parallel x))$, $r_i \leftarrow \{0,1\}^\lambda$, and issues a challenge query to $\mathcal{O}^*_{i,z_i}$ on the input $u_i^* = (x \parallel w \parallel r_i)$ to generate π_i. The resulting proof $(\pi_1, \ldots, \pi_i, \ldots, \pi_\psi)$ is forwarded to $\mathcal{A}$. If $\mathcal{A}$ outputs 0, $\mathcal{B}$ concludes the value returned by the oracle is not the evaluation

of a valid input (P was evaluated on an input which did not contain a valid witness) and we are in $\mathsf{H}_{3,1}$, otherwise we are in $\mathsf{H}_{2,\psi}$. We repeat for $i \in [\psi]$.

□

Finally, observe that $\mathsf{H}_{2,\psi} = \mathsf{H}_3$. By Lemma 4.1.3 and Lemma 4.1.4, for any PPT bounded $\mathcal{A}$ we have that H_0 is indistinguishable from H_3, where the proof is simulated without knowledge of the witness. Any adversary which wins with non-negligible probability implies inverting the PCT-PRF. We conclude that the scheme satisfies zero-knowledge.

□

4.1.2 Comparison with other Works

Since we will focus on a lattice-based instantiation of PCT-PRFs in Construction 6.3.1, we will focus on comparing against NIZKs which can be instantiated from lattice assumptions. The first Designated-Verifier NIZK for NP which can be built from lattice assumptions is due to [52]. Their scheme is built via a black-box transformation from any Public-key Encryption scheme, and a 3-round *interactive* zero-knowledge protocol for honest verifiers (often called a Σ-Protocol). Lattice-based Σ-protocols are still in active development and struggle with efficiency. In general, achieving NIZK schemes from lattice assumptions incurs a tricky balance between soundness and zero-knowledge, which necessitates large amount of parallel repetitions, and rejection sampling in existing strategies. ([48], [13]) It is difficult to compare the computation or communication complexity of their scheme to Construction 4.1.1, without comparing between concrete instantiations. Though, in Chapter 6, we will rely on some heavy machinery in order to construct PCT-PRFs, so it is reasonable to assume that our scheme is less efficient than the most efficient instantiations of [52].

Kim and Wu also constructed as Designated-Prover from lattice assumptions in [47]. Their scheme can either involve a designated verifier, or be made publicly verifiable (but always requires a designated prover). Their work was based on a transformation from the Fully Homomorphic Signature scheme in [39], and any symmetric encryption scheme.

Since their scheme requires very specific tools, it is more reasonable to conjecture performance of their scheme vs. our instantiation. The [47] construction can have its performance potentially improved by restricting attention to evaluation of branching programs (which recognize NC^1), using the evaluation techniques of ([24] , [40]). This is without loss of

generality since 3-SAT is NP-Complete and has an NC^1 witness relation. Since our instantiation is also limited to NC^1, it is more reasonable to theorize about performance comparisons. The constructions are comparable in terms of prover and verification complexity. Any lattice-based schemes which feature some form of constrained circuit evaluation suffer from eventual "correctness degradation", and must have parameters chosen to bound the maximum depth of a circuit to be evaluated. Again, [47] clearly yields smaller public parameters since the correctness degradation incurred can be made much slower.

Both other schemes require that some state be hidden from the verifier to preserve zero-knowledge. In [47], the prover state must be kept secret, and in [52] the trusted authority must keep some state hidden from the verifier (but the prover's parameter can be public). In contrast, it seems that in our construction zero-knowledge is preserved even if the verifier receives everything including any coins used during setup — and we only rely on the designated-verifier restriction to preserve soundness.

Chapter 5

Lattice-based Cryptography

Here, we will recall a minimum set of basic facts and intuitions about lattices and lattice-based cryptography which we will make use of in later constructions.

Notation. We rely on some additional notation for the remaining definitions and constructions. For any $q \in \mathbb{N}$, let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the quotient ring of integers modulo q; we will generally only rely on $\mathbb{Z}_q$ being an additive group. Similarly, we take elements in $\mathbb{Z}_q^{n\times m}$ to be matrices of n-dimensional (column) vectors over $\mathbb{Z}_q$. Matrices $\mathbf{A}$ will be labeled with bold upper-case letters, and vectors $\boldsymbol{v}$ with bold lower-case.

We will focus on definitions in terms of the ℓ_2 (euclidean) and ℓ_∞ norms, denoted as $\|\cdot\|$ and $\|\cdot\|_\infty$ respectively. We extend the definition of norm to matrices in the standard "operator" norm convention such that $\|\mathbf{A}\| = \max_i \|\mathbf{A}\boldsymbol{e}_i\|$ for $\boldsymbol{e}_i$ representing the standard basis vectors with a 1 entry in their i-th position, and 0's elsewhere. So, we have the simplification that $\|\mathbf{A}\|_\infty$ denotes the largest absolute value of any entry in $\mathbf{A}$. We extend concatenation notation to matrices and vectors such that for $\mathbf{V}, \mathbf{W} \in \mathbb{Z}^{n\times k}$, $[\mathbf{V}\|\mathbf{W}] \in \mathbb{Z}^{n\times 2k}$.

We also recall the definition and properties of the Kronecker product, which is the standard tensor product on linear maps defined as follows. For any $i \times j$ matrix $\mathbf{A}$ and $k \times l$ matrix $\mathbf{B}$, the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the $ij \times kl$ matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1j}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{i1}\mathbf{B} & \cdots & a_{ij}\mathbf{B} \end{bmatrix}$$

With the a_{ij} being entries of $\mathbf{A}$. Note that if $\|\mathbf{A}\| = 1$, it follows that $\|\mathbf{A} \otimes \mathbf{B}\| \leq \|\mathbf{B}\|$. In particular, $\|\mathbf{I} \otimes \mathbf{B}\| = \|\mathbf{B}\|$ for any $\mathbf{B}$ (strict equality). We will also rely on the "mixed

product" property of the Kronecker product, which states that for all matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ where the products $\mathbf{AC}$, and $\mathbf{BD}$ are well-defined it holds that:

$$(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$$

In particular we will exploit this in analyses of correctness, and make use of the fact that the Kronecker product is also a multiplicative homomorphism.

5.1 Lattices and Computational Problems

Definition 5.1.1 (Integer Lattices)**.** *An m dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$. A lattice is completely specified by some* ***non-unique*** *set of linearly independent m-dimensional basis vectors $\mathbf{B} = \{b_1, \ldots, b_k\}$. We define the lattice specified by $\mathbf{B}$ as:*

$$\Lambda(\mathbf{B}) = \{\mathbf{B}\boldsymbol{z} \ : \boldsymbol{z} \in \mathbb{Z}^k\}.$$

We call $\mathbf{B}$ a *basis* of Λ, similarly to bases for vector spaces. A lattice $\Lambda(\mathbf{B})$ has rank $k \leq m$ depending on the rank of its basis, e.g. if $k = m$ the lattice is *full-rank*. For every lattice Λ, there also exists a *dual*-lattice Λ^* defined as follows:

$$\Lambda^*(\mathbf{B}) = \{\boldsymbol{v} \in \mathsf{span}(\Lambda) : \langle \boldsymbol{v}, \boldsymbol{x} \rangle \in \mathbb{Z},\ \forall \boldsymbol{x} \in \Lambda(\mathbf{B})\}.$$

Then, $\mathbf{B}^* = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$ is a basis for Λ^* — that is $\Lambda(\mathbf{B}^*) = \Lambda^*(\mathbf{B})$. For any full-rank lattice, we have $\mathbf{B}^* = \mathbf{B}^{-T}$. Finally, it holds that for any lattice Λ, $(\Lambda^*)^* = \Lambda$.

Every lattice has a *fundamental region* defined by the k-dimensional parallelepiped formed by its basis vectors. The volume of the parallelepiped is $|\det \Lambda|$, also called the volume of the lattice. Tiling the fundamental region over $\mathbb{R}^m$ also generates the lattice. So, without loss of generality all operations taking place within the lattice can be done modulo the fundamental region instead.

Next, we state an important definition regarding the set of distinct *shortest* vectors with respect to a ℓ_p norm (discounting the origin point, which is contained in any lattice).

Definition 5.1.2 (Successive Minima)**.** *Let $\mathcal{B}_r$ denote the closed m-dimensional ball with radius r centered at the origin. For a rank k lattice Λ, we define the set $\{\lambda_i\}_{i \in [k]}$ such that for each i:*

$$\lambda_i(\Lambda) = \mathsf{min}\{r : \mathsf{dim}(\mathsf{span}(\Lambda \setminus \{\mathbf{0}\} \cap \mathcal{B}_r)) \geq i\}.$$

Sometimes notation is overloaded and λ_i refers to $\boldsymbol{v} \in \Lambda$ such that$\|\boldsymbol{v}\| = \lambda_i$. Then, the (possibly non-unique) *shortest* vector in any lattice is often called λ_1. A famous theorem by Minkowski states that

$$\lambda_1 \leq \sqrt{k} \cdot |\det \Lambda|^{1/k}.$$

It follows immediately that the set of vectors $\mathbf{B} = \{\lambda_i\}$ forms a basis for Λ. Contrary to this, a set of *any* linearly independent vectors does not necessarily form a basis for the lattice. Since lattices are discrete, the inclusion of a rational scalar multiple of some λ_i means the resulting basis forms some sublattice $\Lambda' \subset \Lambda$ instead. In general, lattice bases are not-unique, as multiplying by any matrix $\mathbf{U}$ with $|\det \mathbf{U}| = 1$ generates a different basis, and preserves the discreteness. This also implies that the volume of the lattice is invariant of the choice of basis. In particular, we may derive a basis $\mathbf{B'}$ for some lattice $\Lambda(\mathbf{B})$ which gives no additional insight regarding the vector λ_1 (or its norm) over the universal bounds.

Next, we briefly define two classical computational problems on lattices. Note that no cryptographic systems have been created whose hardness is based directly on the problems. Instead, we will define a related, but distinct set of problems whose use is currently understood for cryptographic implementations.

Definition 5.1.3 ((Approximate) Shortest Vector Problem (SVP_γ))**.** *Given a random basis* $\mathbf{B}$ *of a lattice* $\Lambda(\mathbf{B})$*, find a* $\gamma \cdot \lambda_1$*-short vector in* Λ*. (That is, find non-zero* $\boldsymbol{v} \in \Lambda$ *such that* $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1$*.)*

In the case of $\gamma = 1$, we have the shortest vector problem (SVP). For $\gamma < \sqrt{2}$, the approximate SVP problem is known to be NP-hard. [49]. On the other hand, for extremely large γ the problem is trivial as any non-zero lattice vector constitutes a valid solution. SVP_γ is known to lie in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$ for $\gamma \approx \sqrt{m}$ [4] (for the remainder of the section we will focus on full-rank lattices). Note that the problem is stated in terms of a *random* basis, since for "nice" bases the problem is trivial. This implies that rewriting a basis in terms of much smaller lattice vectors is difficult as well. Algorithms which accomplish this are called "Basis Reduction" algorithms, which currently are able to solve SVP with sub-exponential approximation factors. Some algorithms are surveyed in Section 2.2 of [53].

Definition 5.1.4 ((Approximate) Closest Vector Problem (CVP_γ))**.** *Given a random basis* $\mathbf{B}$ *of a lattice* $\Lambda(\mathbf{B})$*, and a target point* $\boldsymbol{t} \in \mathsf{span}(\Lambda)$*, find* $\boldsymbol{x} \in \Lambda$ *such that* $\mathsf{dist}(\boldsymbol{t}, \boldsymbol{x}) \leq \gamma \cdot \mathsf{dist}(\boldsymbol{t}, \Lambda)$*. Equivalently, find a point in the lattice whose distance from* $\boldsymbol{t}$ *is no more than* γ *times the distance between* $\boldsymbol{t}$ *and its nearest lattice point.*

It can be shown that $\mathsf{SVP}_\gamma \leq \mathsf{CVP}_\gamma$ for any γ. Intuitively, the shortest vector in a lattice is a non-zero vector which is closest to the origin, so an oracle for CVP which takes the

origin as its input point, outputs the solution to the CVP which also solves SVP — this intuition can be made rigorous for any approximation factor. We can also observe that for any $\gamma \cdot \mathsf{dist}(\boldsymbol{t}, \Lambda) \leq \|\lambda_1\|/2$, the solution to the problem is guaranteed to be unique — this follows from a property sometimes called the *packing distance*.

5.2 Cryptographic Assumptions

When designing cryptographic tools, we are interested in working with problems which are computationally hard in the average-case. We can design such problems on a special family of lattices known as q-ary lattices.

Definition 5.2.1 (q-ary Lattices)**.** *Let $n, m, q \in \mathbb{N}$. For any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, define the following full-rank m dimensional q-ary lattices:*

$$\Lambda^{\perp}(\mathbf{A}) = \{\boldsymbol{z} \in \mathbb{Z}^m : \mathbf{A}\boldsymbol{z} \equiv \mathbf{0} \pmod{q}\}$$
$$\Lambda(\mathbf{A}^T) = \{\boldsymbol{z} \in \mathbb{Z}^m : \exists \boldsymbol{s} \in \mathbb{Z}^n \mid \boldsymbol{z} \equiv \boldsymbol{s}^T\mathbf{A} \pmod{q}\}$$

With the property that $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$. It also holds that:

$$q \cdot \Lambda^{\perp}(\mathbf{A})^* = \Lambda(\mathbf{A}^T)$$

We also consider a generalized definition:

$$\begin{aligned}\Lambda_u^{\perp}(\mathbf{A}) &= \{\boldsymbol{z} \in \mathbb{Z}^m : \mathbf{A}\boldsymbol{z} \equiv \boldsymbol{u} \pmod{q}\} \\ &= \Lambda^{\perp}(\mathbf{A}) + \boldsymbol{x}\end{aligned}$$

which is defined for any syndrome $\boldsymbol{u} \in \mathbb{Z}_q^n$ for which there exists an integral solution to $\mathbf{A}\boldsymbol{x} \equiv \boldsymbol{u} \pmod{q}$. This lattice corresponds to some coset of $\Lambda^{\perp}(\mathbf{A})$ in $\mathbb{Z}^m$ induced by $\boldsymbol{x}$.

Note that in the definition of $\Lambda^{\perp}(\mathbf{A})$, the matrix $\mathbf{A}$ is not the basis of the lattice — rather the basis is defined by some basis for $\mathsf{Ker}(\mathbf{A})$. Next, we state a generalization of the Leftover Hash Lemma [44] necessary for establishing later definitions.

Lemma 5.2.1 (Lattice Leftover Hash Lemma ([56] Claim. 5.3, [35] Lemma 5.1))**.** *Let $n, m, q \in \mathbb{N}$. For a constant $c > 1$ and $m \geq cn \log q$, let $\epsilon = q^{-(c-1)n/4}$. Then with probability $1 - \epsilon$, for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^m, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n$, we have*

$$\Delta(\mathbf{A}\boldsymbol{x} \bmod q, \boldsymbol{u}) < \epsilon.$$

In particular, it follows from Lemma 5.2.1 that for some $m > 2n \log q$ and any $\boldsymbol{u} \in \mathbb{Z}_q^n$, there exists a $\boldsymbol{x} \in \{0,1\}^m$ such that $\mathbf{A}\boldsymbol{x} \equiv \boldsymbol{u} \pmod q$. That is, any syndrome of $\Lambda_u^\perp(\mathbf{A})$ is well-defined. Now, we can define our first computational assumption.

Definition 5.2.2 (Short Integer Solution ($\mathsf{SIS}_{n,q,m,\beta}$))**.** *Given $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and a bound β, find a non-zero vector $\boldsymbol{z} \in \mathbb{Z}_q^m$ with norm $\|\boldsymbol{z}\| \leq \beta$ such that*

$$\mathbf{A}\boldsymbol{z} \equiv \mathbf{0} \pmod q.$$

For the problem to admit a solution in general, we require $\beta > \sqrt{n(\lfloor \log q \rfloor + 1)}$. We can also notice that the problem implies a collision resistance function $\mathbf{A}\boldsymbol{x} \bmod q$ for any small $\boldsymbol{x}$ (e.g. in $\{0,1\}^m$) since any collision immediately yields a short solution to 0. We will rely on this fact in the next section.

Theorem 5.2.1. *For any $m = \mathsf{poly}(n)$, $\beta > 0$, and sufficiently large $q \geq \beta \cdot \mathsf{poly}(n)$, $\mathsf{GapSVP}_\gamma \leq \mathsf{SIS}_{n,q,m,\beta}$ for some $\gamma = \beta \cdot \mathsf{poly}(n)$, where the reduction implies solving GapSVP_γ on **any** n-dimensional lattice.*

The problem GapSVP is a decisional version of Definition 5.1.3 (i.e. decide whether the $\|\lambda_1\|$ is less than or greater than some bound) which has similar hardness to SVP across the spectrum of approximation factors. In [35], it was shown that the bounds $\mathsf{poly}(n)$ for both the size of the modulus and the approximation factor could be set to $\tilde{O}(\sqrt{n})$. Improved bounds were established in later works ([53]). Note that SIS resembles solving a shortest-vector problem on the lattice $\Lambda^\perp(\mathbf{A})$, and the hardness of the problem decreases as m increases.

Another remarkable aspect of the theorem statement is that solving an *average-case* instance of SIS (e.g. a problem on a *special* m-dimensional lattice) implies solving a different problem on *any* n-dimensional lattice, i.e. in the *worst-case*.

Next, we will define a distribution necessary for defining the rest of the properties we will need. Going forward, all problems and constructions will be presented, and proven secure assuming the hardness of *some* lattice problem with a certain approximation factor.

Definition 5.2.3 (Discrete Gaussian Distributions Over Lattices [35])**.** *Define the Gaussian function on $\mathbb{R}^n$ centered at $\boldsymbol{c}$ with standard deviation $\sigma > 0$ as:*

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \rho_{\sigma,\boldsymbol{c}}(x) = \rho_{\boldsymbol{c}}(x/\sigma) = e^{-\pi \|\boldsymbol{x}-\boldsymbol{c}\|^2/\sigma^2}$$

If $\boldsymbol{c}$ is omitted, it is assumed to be $\mathbf{0}$. Then, the Discrete Gaussian distribution over a lattice Λ is the probability measure assigned to all points $\boldsymbol{x} \in \Lambda$:

$$D_{\Lambda,\sigma,\boldsymbol{c}} = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{\Lambda})}$$

So, sampling from $D_{\Lambda,\sigma,\boldsymbol{c}}$ produces discrete values with respect to a Gaussian probability mass function with parameter σ. In particular, the mass assigned to any $\boldsymbol{x} \notin \Lambda + \boldsymbol{c}$ is 0. We will sometimes use the notation above in a compact form when sampling matrices, e.g. $D_{\mathbb{Z},\sigma}^{k\times m} = (D_{\mathbb{Z}^k,\sigma})^m$. Note that the use of $\mathbb{Z}^k$ is no different than the definition above since $\mathbb{Z}^k = \Lambda(\mathbf{I}_k)$ is the k-dimensional standard integer lattice for any $k \in \mathbb{N}$.

Next, we state a celebrated result by Regev [56] which defined a generalization of the "Learning Parity with Noise" problem. We state the decisional version of the problem.

Definition 5.2.4 ((Decisional) Learning with Errors ($\mathsf{LWE}_{n,q,\chi}$))**.** *Let $n = \mathsf{poly}(\lambda), q \in \mathbb{N}$, and $\chi = D_{\mathbb{Z},\sigma} : \sigma \geq 2\sqrt{n}$. Then for any $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n$, private $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, and $e \leftarrow \chi$, distinguish between the following two distributions:*

$$(\boldsymbol{a}, \boldsymbol{u}), (\boldsymbol{a}, \langle \boldsymbol{s}, \boldsymbol{a} \rangle + e \bmod q)$$

with probability significantly better than $1/2 + \mathsf{negl}(\lambda)$.

Regev showed that solving LWE in the average-case implies solving GapSVP_γ in the worst-case via a *quantum* algorithm. In particular, if we *fix* $\boldsymbol{s}$, and for any $m = \mathsf{poly}(n)$ sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, and $\boldsymbol{e}^T \leftarrow \chi^m$, then we can define $\mathsf{LWE}_{n,m,q,\chi}$ as to claim:

$$(\mathbf{A}, \boldsymbol{b} = \boldsymbol{s}^T\mathbf{A} + \boldsymbol{e}^T \bmod q) \approx_c (\mathbf{A}, \boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_q^m)$$

Theorem 5.2.2 ([56])**.** *Let $n = \mathsf{poly}(n)$, $q \leq 2^{\mathsf{poly}(n)}$, and $\chi = D_{\mathbb{Z},\sigma}$ with $\sigma \geq 2\sqrt{n}$. For any $m = \mathsf{poly}(m)$ and $\gamma = \tilde{O}(nq/\sigma)$, it holds that:*

$$\mathsf{GapSVP}_\gamma \leq_Q \mathsf{LWE}_{n,m,q,\chi}$$

We can think of $\mathsf{LWE}_{n,m,q,\chi}$ as solving a variant of CVP_γ with a distance guarantee (called "Bounded Distance Decoding"), given the point $\boldsymbol{t} = \boldsymbol{s}^T\mathbf{A} + \boldsymbol{e}^T \bmod q$ near the lattice $\Lambda(\mathbf{A}^T)$. We typically refer to $\boldsymbol{s}$ as the *secret* term, and $\boldsymbol{e}$ as the *error* or *noise* term (which is also a secret in the problems above). It was shown in [35] that the family of lattices

$\Lambda(\mathbf{A}^T)$ defined by any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ has the particularly nice feature that $\|\lambda_1\|_\infty \leq q/4$. This implies that if an LWE error term is bounded by $q/4$, we can unambiguously recover a distinct lattice point. Generalizations of this fact form the basis of most lattice-based encryption schemes; we will also make use of this property in our constructions.

While simply rounding the output of $\rho_{\sigma,c}$ may produce an output from the wrong distribution, [35] showed it is also possible to efficiently sample within negligible statistical distance of a discrete Gaussian distribution over any lattice, and any coset. This algorithm was improved in [21] to sample from the distribution exactly. This means that LWE-based primitives can also actually be implemented in practice on physical devices since we can handle the simple case of sampling over $\mathbb{Z}$. (Otherwise we may have a separation between what can be proved, and what can be implemented)

Lemma 5.2.2 ([21])**.** *Let* $\mathbf{B}$ *be a basis for a lattice* $\Lambda = \Lambda(\mathbf{B})$*, and* $\tilde{\mathbf{B}}$ *be it's Gram-Schmidt orthogonalization. Then there is a* PPT *algorithm which samples from* $D_{\Lambda,\sigma,\boldsymbol{c}}$ *for any* $\boldsymbol{c} \in \mathbb{R}^m$*, and* $\sigma > \|\tilde{\mathbf{B}}\| \cdot \sqrt{\ln(2m+4)/\pi}$*.*

Now, we will bound the norm of samples drawn from discrete Gaussian distributions which will be necessary in later analysis. This will also justify that it is quite easy to sample from a distribution χ which is wide enough to satisfy the requirements of Definition 5.2.4 while preserving unique decoding of vectors in $\Lambda(\mathbf{A}^T)$.

First, we introduce a quantity η_ϵ known as the *smoothing parameter* [51], which is one of the most important concepts in lattice-based cryptography. This is the minimal parameter σ such that the expectation of the discrete Gaussian $D_{\Lambda,\sigma,\boldsymbol{c}}$ is statistically close to that of the *continuous* m-dimensional Gaussian centered at $\boldsymbol{c}$ with the same σ. As a consequence, [49] shows that adding Gaussian noise with parameter $\sigma > \eta_\epsilon$ to any lattice point results in a point within statistical distance 2ϵ from uniform over $\mathbb{R}^m$.

The quantity η_ϵ is defined such that a continuous Gaussian over the dual lattice has most of its mass centered around the origin. The reason for this definition is complex, but a nice intuition is that applying a Fourier transform to a "narrow" continuous Gaussian distribution over Λ^* corresponds to a "wide" (and hence close to uniform) *discrete* distribution over Λ. These properties end up being crucial to the reductions in [56], and provides a different perspective on why *discrete gaussian* noise is present in Definition 5.2.4.

Lemma 5.2.3 (Smoothing Parameter)**.** *For an* m*-dimensional lattice* Λ *and* $\epsilon > 0$*, the smoothing parameter* $\eta_\epsilon(\Lambda)$ *is the smallest* σ *such that* $\rho_{1/\sigma}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$*.*

Lemma 5.2.4 ([35])**.** *For any* m*-dimensional lattice* Λ *and* $\epsilon > 0$*, we have*

$$\eta_\epsilon(\Lambda) \leq \|\tilde{\mathbf{B}}_{min}\| \cdot \sqrt{\ln(2m(1+1/\epsilon))/\pi},$$

where $\mathbf{B}_{min} = \{\lambda_1, \ldots, \lambda_m\}$ *is the* ***minimum*** *basis of* Λ*, and* $\|\tilde{\mathbf{B}}\|$ *denotes the norm of its Gram-Schmidt orthogonalization.*

We can then bound the norm of any discrete Gaussian sample.

Lemma 5.2.5 ([51])**.** *For any* m*-dimensional lattice* Λ*,* $\boldsymbol{c} \in \mathsf{span}(\Lambda)$*,* $\epsilon \in (0, 1)$*, and* $\sigma \geq \eta_\epsilon(\Lambda)$*, we have*

$$\Pr_{\boldsymbol{x} \leftarrow D_{\Lambda,\sigma,\boldsymbol{c}}} \left[\|\boldsymbol{x} - \boldsymbol{c}\| > \sigma\sqrt{m} \right] \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-m}.$$

If $\boldsymbol{c} = \mathbf{0}$*, the bound holds for any* $\sigma > 0$*, with* $\epsilon = 0$*.*

Now, we will define useful generalizations of the LWE assumption. We can easily extend the definition to matrices by noticing each row of any $\mathbf{B} = \mathbf{SA} + \mathbf{E} \bmod q$ takes the form $\boldsymbol{s}_i^T\mathbf{A} + \boldsymbol{e}_i^T \bmod q$. So, provided we take $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$ and $\mathbf{E} \leftarrow \chi^{n \times m}$ we can demonstrate hardness through a simple hybrid argument in which each row of $\mathbf{B}$ is replaced with a uniform element in $\mathbb{Z}_q^m$ by invoking $\mathsf{LWE}_{n,m,q,\chi}$.

In [8] it was shown that LWE remains hard if the secrets are sampled from the error distribution χ (e.g. the secrets have small norm) (with a small loss on the number of samples m), this is called the "Normal Form" LWE. Boneh et al. [16] showed the problem remains hard if the public matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}$ has *low-norm* entries provided the dimension n is increased to $O(n \log q)$ and the secrets are drawn uniformly over $\mathbb{Z}_q^n$. Chen et al. [31] generalized these two ideas and showed that the roles of the $\mathbf{S}$ and $\mathbf{A}$ terms could also be swapped (i.e. $\mathbf{A}$ becomes the secret component). This holds even if $\mathbf{S}$ has small norm, as long as $\mathbf{A}$ remains uniformly distributed and dimensions are chosen appropriately.

We will make use of the [8] and [16] version of LWE in this work, so to distinguish them we will adopt the notation of [31] and specify the problem with the subscripts $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$ for a secret distribution θ, a public distribution π, and an error distribution χ. For brevity, we may drop subscripts when not necessary, or obvious from the context.

5.3 Lattice Trapdoor Techniques

The techniques established in the previous section show we can sample from discrete Gaussian distributions over any lattice, with the norm depending on the "quality" of the basis (e.g. $\|\tilde{\mathbf{B}}\|$). In particular, we sample short vectors from $\Lambda^\perp(\mathbf{A})$ if we pick some small basis for $\mathsf{Ker}(\mathbf{A})$ and derive $\mathbf{A}$ later, which lets us solve $\mathsf{SIS}_{n,q,m,\beta}$ for β as stated in Lemma 5.2.5. However, this requires some careful consideration. The average-case hardness of SIS is

on *average* over all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. The procedure described above could instead produce pathological $\mathbf{A}$ for which the problem is not difficult. Moreover, preimages output by the algorithm could leak information about the private $\mathbf{B}$.

What we would like is to be able to sample $\mathbf{A}$ which is close to uniform over $\mathbb{Z}^{n \times m}$ together with a basis which lets us sample short solutions. If $\mathbf{A}$ is indeed close to uniform, SIS implies it should be difficult to find a basis which allows sampling solutions with norm less than β. This leads us to the construction of a *trapdoor* function, for which it is difficult to find preimages, but easy to verify congruences $\mathbf{A}\boldsymbol{x} \bmod q$ — in which case we can consider the "short" basis a trapdoor $\mathbf{T}_\mathbf{A}$ for $\mathbf{A}$.

In particular, we would like to sample short solutions from any coset $\boldsymbol{u}$ of $\Lambda_{\boldsymbol{u}}^{\perp}(\mathbf{A})$ since this is strictly more useful than sampling preimages of 0. For example, such a scenario immediately yields a signature scheme in the random oracle model where $\boldsymbol{u} = H(m)$. However, in order to make security claims we first need to define an inhomogenous variant of SIS to relate any scheme to a computational assumption.

Definition 5.3.1 (Inhomogenous Short Integer Solution ($\mathsf{I\text{-}SIS}_{n,q,m,\beta}$))**.** *Given $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a bound β, find a vector $\boldsymbol{x} \in \mathbb{Z}_q^m$ with norm $\|\boldsymbol{x}\| \leq \beta$ such that:*

$$\mathbf{A}\boldsymbol{x} \equiv \boldsymbol{u} \pmod{q}$$

It follows immediately that $\mathsf{SIS}_{n,q,m,2\beta} \leq \mathsf{I\text{-}SIS}_{n,q,m,\beta}$ since any collision $\boldsymbol{x}, \boldsymbol{x}' : \mathbf{A}\boldsymbol{x} \equiv \mathbf{A}\boldsymbol{x}' \pmod{q}$ implies $\mathbf{A}(\boldsymbol{x} - \boldsymbol{x}') \equiv \mathbf{0} \pmod{q}$ which yields a solution $(\boldsymbol{x} - \boldsymbol{x}')$ to SIS with norm $\|\boldsymbol{x} - \boldsymbol{x}'\| \leq 2\beta$. In practice, any parameters must be chosen so that not only can we sample preimages for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with norm β, but $\mathsf{SIS}_{n,q,m,2\beta}$ is hard with respect to a security parameter λ.

We now state the minimum set of requirements for achieving lattice trapdoors. All of the techniques in the literature satisfy the requirements sketched above in terms of producing a random looking $\mathbf{A}$ and not leaking information about the trapdoor. The best performing techniques are due to [50], which do not follow the "short basis" approach but are limited to q-ary lattices, whereas the strategies in [35] work for any lattice.

Lemma 5.3.1 (Lattice Trapdoors and Preimage Sampling ([5], [35], [50]))**.** *Let $n, q \in \mathbb{N}$, and $m = O(n \log q)$. Then there exists a PPT algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ which outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $(\mathbf{A}, \mathbf{T}_\mathbf{A})$, such that $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$. Additionally, there is a PPT algorithm $\mathsf{PreSample}(\mathbf{T}_\mathbf{A}, \boldsymbol{u}, \sigma)$ which for any $\sigma \geq O(\sqrt{n \log q})$ and any $\boldsymbol{u} \in \mathbb{Z}_q^n$ outputs a sample $\boldsymbol{x} \in \mathbb{Z}^m$ from $D_{\Lambda^{\perp}(\mathbf{A}),\boldsymbol{u},\sigma}$ such that $\mathbf{A}\boldsymbol{x} \equiv \boldsymbol{u} \pmod{q}$.*

For convenience we will sometimes abbreviate the output $\boldsymbol{d}$ of $\mathsf{PreSample}(\mathbf{T}_{\mathbf{A}}, \boldsymbol{u}, \sigma)$ as $\boldsymbol{d} = \mathbf{A}^{-1}(\boldsymbol{u}, \sigma)$, omitting the standard deviation parameter outside of proofs to simplify exposition. Of course, $\mathbf{A}$ is not invertible, but it is the case that $\mathbf{A}\boldsymbol{d} \equiv \boldsymbol{u} \pmod{q}$. The definition extends immediately to sampling preimages of any $n \times k$ matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$.

Remark 5.3.1. Some works which focus on $\|\cdot\|_\infty$ often write the output of $\mathsf{PreSample}$ as being $O(n\sqrt{\log q})$ bounded, rather than $(n \log q)^{-1/2} \cdot \|\sigma \cdot \sqrt{m}\|_\infty \leq (\sqrt{n \log q})^2$. We can restrict $\mathsf{PreSample}$ to only outputting samples with this bound. A similar remark is made briefly in [34] with the qualifier that this does not impact the resulting distribution by a significant factor. Of course, we could always treat any ℓ_2 norm values as worst-case bound in ℓ_∞ analyses instead, which results in more well distributed values, but makes analysis more nuanced.

Lemma 5.3.2 ([27], [35])**.** *For any* $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $\sigma > 2\sqrt{n \log q}$ *the following holds with overwhelmingly high probability:*

$$(\mathbf{A}, \boldsymbol{x}, \boldsymbol{y} : \boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_q^n, \boldsymbol{x} = \mathbf{A}^{-1}(\boldsymbol{y}, \sigma)) \approx_s (\mathbf{A}, \boldsymbol{x}, \boldsymbol{y} : \boldsymbol{y} = \mathbf{A}\boldsymbol{x}, \boldsymbol{x} \leftarrow D_{\mathbb{Z}^m, \sigma})$$

Proof. The statement follows immediately from Lemmas 5.2, 5.3 and Corollary 5.4 in [35], since the matrix $\mathbf{A}$ output by $\mathsf{TrapGen}(1^n, 1^m, q)$ is statistically close to uniform for sufficiently large $m \geq 2n \log q$, the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$ with overwhelmingly high probability, and $\sigma = \omega(\sqrt{\log m}) \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$.

□

Chapter 6

Construction from Learning with Errors

6.1 Lattice-based PRF Fundamentals

Various lattice PRFs have been studied since the first construction by Bannerjee et al. in [10]. Many of the constructions are syntactically similar, but depending on choices of parameters and distributions, must be based on widely different security assumptions. We will review simplified descriptions of the underlying structures, and outline the challenges faced when trying to build a PRF which is range-testable and privately constrained from lattice assumptions.

The first intuition in constructing lattice PRFs is that the high-order bits of any LWE sample $\boldsymbol{b} = \boldsymbol{s}^T\mathbf{A} + \boldsymbol{e}^T \bmod q$ are unaffected by $\boldsymbol{e}$, given that the error is drawn from some B-bounded distribution. However, LWE says that *the entirety* of the resulting product is indistinguishable from random — including those bits which are not affected by the error. With this observation, we could consider *rounding* the product $\boldsymbol{s}^T\mathbf{A} \bmod q$ instead. Then, as long as the amount of low-order bits dropped by rounding exceeds the maximum size of any error $\boldsymbol{e}$ with high probability, rounding should produce an identical output instead of sampling an error. This gives a *deterministic* function which is also pseudorandom, and we have the main ingredients for a PRF.

In particular, for a rounding function $\lfloor\cdot\rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ defined as:

$$\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p,$$

then for any $q/p > B \cdot n^{\omega(1)}$, we have

$$\lfloor \boldsymbol{s}^T \mathbf{A} + \boldsymbol{e}^T \rceil_p = \lfloor \boldsymbol{s}^T \mathbf{A} \rceil_p.$$

Bannerjee et al. also defined a related problem called *Learning with Rounding* (LWR), which is essentially the right side of the expression above. The problem asks to distinguish noisy inner products where the noise is introduced only by rounding. They showed that for an appropriate amount of rounding specified by p, $\mathsf{LWR}_{n,q,p}$ is equally hard as $\mathsf{LWE}_{n,q,\chi}$. The original construction by Bannerjee et al. [10] took the form:

$$\mathsf{F}_{\mathbf{A},\{\mathbf{S}_{i,b}\}}(x) = \left\lfloor \prod_{i \in |x|} \mathbf{S}_{i,x_i} \cdot \mathbf{A} \right\rceil_p \tag{6.1.1}$$

For a secret key consisting of $\{\mathbf{S}_{i,b} \leftarrow \chi^{n \times n}\}_{i \in |x|, b \in \{0,1\}}$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and security based on the normal form LWE assumption.

A subsequent work by Boneh et al. [16] showed that an alternative construction also possessed key-homomorphic properties, where $\mathbf{A}_0, \mathbf{A}_1 \in \{0,1\}^{m \times m}$ are global *public low-norm* matrices, and the key is a vector $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^m$:

$$\mathsf{F}_{\boldsymbol{s}}(x) = \left\lfloor \boldsymbol{s}^T \cdot \prod_{i \in |x|} \mathbf{A}_{x_i} \right\rceil_p \tag{6.1.2}$$

$$= \lfloor \boldsymbol{s}^T \cdot \mathbf{A}_x \rceil_p \tag{6.1.3}$$

A major difference in this scheme was that key size was reduced dramatically, and that proving security required a different LWE assumption with *small normed, higher dimension public* matrices (Definition 5.2.4).

Towards Constraints and Testability. We will use the observation that any PRF structured like Eq. (6.1.1) can be made testable by sampling $\mathbf{A}$ together with trapdoor. The intuition is that the underlying geometric behaviour of the PRF produces points which are close to a lattice defined by the secret key and the input. The trapdoor lets us test this closeness as long as the distance (error) is sufficiently bounded. So, a possible strategy is to try to construct a Privately Constrained PRF which corresponds to Eq. (6.1.1), and attempt to introduce testability using the trapdoor.

The first step can be accomplished using the "GGH15" graph-induced encoding technique of [34], and was first shown in [27]. However, their construction is also not testable as described. Informally, the construction preserves the structure of Eq. (6.1.1) aside from a lossy map which is applied to the $\mathbf{A}$ term of the secret key. The map they choose in their construction is a considerable barrier to testability. Though, *some* lossy map is required to make the construction secure, so we will consider an alternate map which does not have this problem. We will explore this in detail in Section 7.5 after introducing some necessary definitions.

6.2 GGH15 Encodings and Branching Programs

In [34], it was observed that the homomorphic capabilities of LWE could be creatively combined with lattice trapdoors to encode constrained walks along directed acyclic graphs. Extensions of this idea were used to propose candidate multi-linear maps, non-interactive multi-party key exchange, and other constructions.

We start with a simple example to sketch the intuitions. The example will does not propose any particular cryptosystem, but serves as a more concrete description of how lattice techniques can be used to encode a constrained walk on a directed graph. This concept will be crucial to realizing actual applications. Consider the simple directed graph $G(V, E)$ with vertices $v_{i,j}$ $V = \{v_0, v_{1,0}, v_{1,1}, \ldots, v_{n,1}\}$ below.

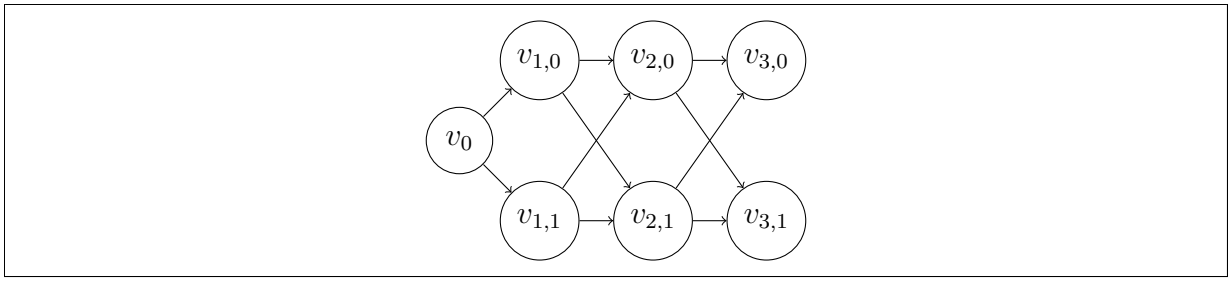


Figure 6.1: A simple directed graph

Suppose we associate a matrix and trapdoor $(\mathbf{A}_{i,j}, \mathbf{T}_{\mathbf{A}_{i,j}})$ with each vertex (and $\mathbf{A}_0$ for v_0). We can use lattice techniques to allow any party to create an LWE sample corresponding to the last vertex in a constrained walks as follows:

1. Select a path from v_0, $P = \{b_1, b_2, b_3 \mid \mathbf{A}_{i,b_i} \to \mathbf{A}_{i+1,b_{i+1}}\}$

2. Sample $\mathbf{S}_i \leftarrow \chi^{n\times n}, \mathbf{E}_i \leftarrow \chi^{n\times m}$
3. Compute $\mathbf{B}_i = \mathbf{S}_i \cdot \mathbf{A}_{i+1,b_{i+1}} + \mathbf{E}_i$
4. Sample $\mathbf{D}_i \leftarrow \mathbf{A}_{i,b_i}^{-1}(\mathbf{B}_i) \in \mathbb{Z}^{m\times m}$
5. Output $\widetilde{P} = (\mathbf{A}_0, \{\mathbf{D}_i\}_{i\in[3]})$

Then,

$$\mathbf{A}_0 \prod_{i=1}^{3} \mathbf{D}_i \approx \left(\prod_{i=1}^{3} \mathbf{S}_i\right) \mathbf{A}_{3,b_3} \tag{6.2.1}$$

We will analyze the exact structure of such products in the next section. Informally, the approximation holds because preimages output by lattice trapdoors have low norm, and we can rely on the hardness of LWE with both small *secrets* and error terms. If all the matrices $\mathbf{A}_{i,b}$ are public, we have the following concepts:

1. By assumption only the owner of the trapdoors $\mathbf{T}_{\mathbf{A}_{i,j}}$ should be able to sample preimages.
2. Since each $\mathbf{B}_i$ is an LWE sample, then each $\mathbf{A}_{i,b}\mathbf{D}_i = \mathbf{S}_i \cdot \mathbf{A}_{i+1,b_{i+1}} + \mathbf{E}_{i,b}$ is also an LWE sample for matching i, b
3. Every intermediate product resembles an LWE with a more complex error term.

Of course, given the set of vertices anyone can produce an LWE sample, but producing samples under the output vertex matrices in the manner above implies knowledge of the trapdoors. Considering these properties, it is tempting to claim that the entire path is hidden given the encoding.

However, until recently it was unknown how to obtain any positive security results for similar encodings. Depending on the goals of the construction, components which may need to stay hidden for security may need to be made public to argue distributional changes are indistinguishable. In fact, many early candidate constructions have been demonstrated to be insecure ([32], [30]). It was shown only later, in [27], [42], [58] how to use these techniques in a secure manner for different applications than those originally proposed by [34].

The GGH15 encodings briefly sketched above are more interesting when combined together with *branching programs* (via Barrington's Theorem). Branching programs can represent all circuits in the class NC^1, and can be evaluated by performing a set of ordered state transitions which is representable by a walk on a graph of the form in Fig. 6.1. In such a setting, the encoder can publish encodings corresponding to all valid state transitions $\{\mathbf{D}_{i,b}\}_{b\in\{0,1\}}$. Here, the subscripts b depend on the i-th input bit of a circuit $C \in \mathsf{NC}^1$.

Restricting our attention to a public graph with a fixed topology, we wish to enable an evaluator to perform all possible connected walks. This allows them to evaluate a program on any possible input. Assuming LWE, we will be able to prove that the program C, and its output $C(x)$ are hidden. The encoding strategy will differ from the sketch in that we will not index the matrices for each vertex by b, but instead encode two possible edges in the preimages $\mathbf{D}_{i,b}$ by introducing structure. We will structure encodings such that a directed walk on a graph like the one above induces a directed walk on a binary tree (we can view branching programs as binary decision trees). The encoding strategy will seemingly deviate from older analyses of LWE, and require some more careful security analysis, but this will be easily remedied ([27], [42], [58], [31]). Next, we will describe the details and formally define these properties.

Definition 6.2.1 (Permutation Branching Programs)**.** *A width w, length L permutation branching program $\{\mathsf{BP}(C)\}$ for some circuit $C : \{0,1\}^{\ell_{in}} \to \{0,1\}$ consists of an ordered sequences of $w \times w$ permutation matrices $\{\mathbf{P}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[L],b\in\{0,1\}}$, an index to input map $\iota : [L] \to [\ell_{in}]$, and some fixed permutation $\mathbf{P}^* \in \{0,1\}^{w\times w} \setminus \mathbf{I}_w$. A permutation branching program is evaluated as follows:*

$$\mathsf{BP}(C)(x) = \begin{cases} 1 & \text{if } \prod_{i\in[L]} \mathbf{P}_{i,x_{\iota(i)}} = \mathbf{I}_w \\ 0 & \text{if } \prod_{i\in[L]} \mathbf{P}_{i,x_{\iota(i)}} = \mathbf{P}^* \\ \bot & \text{else} \end{cases}$$

Theorem 6.2.1 (Barrington's Theorem ([12]))**.** *For $d \in \mathbb{N}$, and all depth-d circuits $\mathcal{C} \subset \mathsf{NC}^1$, represented as functions $C : \{0,1\}^{\ell_{in}} \to \{0,1\}$ there exists a set of width $w = 5$, length $L = 4^d$ branching programs $\{\mathsf{BP}(C)\}_{C\in\mathcal{C}}$ with the same index-to-input map $\iota : [L] \to [\ell_{in}]$, such that: $\mathsf{BP}(C) = (\{\mathbf{P}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[L],b\in\{0,1\}}, \mathbf{P}^*, \iota)$.*

The definition above can be generalized to non-permutation matrices, which can significantly improve efficiency but requires dramatically different constructions and proofs [31]. The property that all branching programs resulting from Theorem 6.2.1 have the same index-to-input map is of great utility for proving indistinguishability of circuits. We call

these *oblivious* branching programs. Furthermore, any branching program can be made to have the simple map $\iota(i) = i \bmod \ell_{in}$ at the cost of increasing the length of the branching program.

Theorem 6.2.2 (GGH15 Evaluation Correctness)**.** *Let* $n, m, w, L \in \mathbb{N}$, $t = wn, q \leq 2^{\mathsf{poly}(n)}, m > \Omega(t \log q)$, *and let* $\{\mathbf{A}_i \in \mathbb{Z}_q^{t \times m}\}_{i \in [L+1]}$, $\{\mathbf{S}_{i,b} \in \mathbb{Z}_q^{n \times n}, \mathbf{E}_{i,b} \in \mathbb{Z}_q^{t \times m}, \mathbf{D}_{i,b} \in \mathbb{Z}_q^{m \times m}\}_{i \in [L], b \in \{0,1\}}$, *with* $\beta^* = \max_{i,b}\{\|\mathbf{S}_{i,b}\|_\infty, \|\mathbf{E}_{i,b}\|_\infty, \|\mathbf{D}_{i,b}\|_\infty\}$. *Let* $\hat{\mathbf{S}}_{i,b} = \mathbf{I}_w \otimes \mathbf{S}_{i,b}$. *Then, there exists* $q \in \mathbb{N}$ *such that for all* $x \in \{0,1\}^L$*:*

$$\mathbf{A}_1 \prod_{i=1}^{L} \mathbf{D}_{i,x_i} \approx \left(\prod_{i=1}^{L} \hat{\mathbf{S}}_{i,x_i} \right) \mathbf{A}_{L+1} \pmod q,$$

where for all i, b*:* $\mathbf{D}_{i,b} = \mathbf{A}_i^{-1}(\hat{\mathbf{S}}_{i,b} \cdot \mathbf{A}_{i+1} + \mathbf{E}_{i,b})$.

Proof. Consider the expanded terms below, where the relation holds modulo q:

$$\mathbf{A}_1 \prod_{i=1}^{L} \mathbf{D}_{i,x_i} \equiv \left(\prod_{i=1}^{L} \hat{\mathbf{S}}_{i,x_i} \right) \mathbf{A}_{L+1} + \underbrace{\sum_{i=1}^{L} \left(\prod_{j=1}^{i-1} \hat{\mathbf{S}}_{i,x_i} \cdot \mathbf{E}_{i,x_i} \cdot \prod_{k=i+1}^{L} \mathbf{D}_{k,x_k} \right)}_{\mathbf{E}_x} \tag{6.2.2}$$

$$\approx_\epsilon \left(\prod_{i=1}^{L} \hat{\mathbf{S}}_{i,x_i} \right) \mathbf{A}_{L+1} \tag{6.2.3}$$

Suppose $\beta = \Omega(t\sqrt{\log q})$. Then, a loose upper bound on the error term above is: $\|\mathbf{E}_x\|_\infty = \beta^* \leq \beta(2m\beta)^{L-1}$. So, for $q > \beta^*$, Eq. (6.2.3) holds with precision $\epsilon = \beta^*/q$, which can be easily satisfied if both (β, m) have at most logarithmic dependence on q.

□

Note that above the secret term $\mathbf{I}_w \otimes \mathbf{S}_{i,b}$ simply maps the secret $\mathbf{S}_{i,b}$ to each n rows of an $nw \times m$ matrix, so we can simply view this as "giving out more samples". Recall from Definition 5.2.4 that as long as the number of samples remains polynomial this doesn't impact hardness by any relevant factor.

Corollary 6.2.1. *For β^*, q as in Theorem 6.2.2, for any $p \in \mathbb{Z}$ such that $p < \beta^*/q$, we have*

$$\mathbf{A}_1 \prod_{i=1}^{L} \mathbf{D}_{i,x_i} \equiv \left(\prod_{i=1}^{L} \hat{\mathbf{S}}_{i,x_i} \right) \mathbf{A}_{L+1} \pmod{p}.$$

Now, we can describe how to embed branching programs into a GGH15 encoding by exploiting multiplicative homomorphism and adding some additional structure. Consider when each $\hat{\mathbf{S}}$ term is instead written as: $\hat{\mathbf{S}}_{i,x_i} = \mathbf{P}_{i,b} \otimes \mathbf{S}_{i,x_i}$, where $\mathbf{P}_{i,b}$ is some $w \times w$ permutation matrix[6]. Then, a GGH15 encoding of a permutation branching program is a set of encodings of permutation matrices, and has preimages of the following form:

$$\mathbf{A}_i \mathbf{D}_{i,b} = (\mathbf{P}_{i,b} \otimes \mathbf{S}_{i,b}) \cdot (\mathbf{I}_{nw} \cdot \mathbf{A}_{i+1}) + \mathbf{E}_{i,b} \tag{6.2.4}$$

The correctness analysis above is preserved, since multiplying by a permutation matrix has no effect on the ℓ_∞ norm. So, it suffices to prove that the resulting evaluation produces a valid encoding of a branching program output, we will show this below. Informally, this works because of the homomorphic properties of the encodings. Since we will only encode branching programs, which are oblivious — an evaluator can simply "pretend" the square matrices $\{\mathbf{D}_{i,b}\}$ are the choices of permutation matrices, and compute a subset product which depends on their input without having to know the underlying circuit description.

Lemma 6.2.1 (GGH15 Evaluation Correctness for Branching Programs (follows from [34], [27]))**.** *Let $\mathsf{BP}(C) = \{\mathbf{P}_{i,b}\}_{i \in [L], b \in \{0,1\}}$ be the length L, width w branching program representing a circuit $C \in \mathsf{NC}^1$ of the form $C : \{0,1\}^{\ell_{in}} \to \{0,1\}$. Also, let $\mathbf{A}_1, \{\mathbf{D}_{i,b}\}_{i \in [L], b \in \{0,1\}}$ be a GGH15 encoding of $\mathsf{BP}(C)$. Then, for any $x \in \{0,1\}^{\ell_{in}}$ the GGH15 evaluation process produces an encoding of $C(x) \in \{\mathbf{I}_w, \mathbf{P}^*\}$.*

Proof. Let $\hat{\mathbf{S}}_{i,b} = \mathbf{P}_{i,b} \otimes \mathbf{S}_{i,b}$, and $\mathbf{D}_{i,b} = \mathbf{A}_i^{-1}(\hat{\mathbf{S}}_{i,b} \cdot \mathbf{A}_{i+1} + \mathbf{E}_{i,b})$ as in Theorem 6.2.2.

Then for any input $x \in \{0,1\}^{\ell_{in}}$, by Eq. (6.2.3) and multiplicative homomorphism of the Kronecker product:

[6] In fact, this format is not restricted to the Kronecker product, but can also be applied to any multiplicative homomorphism $\gamma : \mathbb{Z}^{w \times w} \times \mathbb{Z}^{n \times n} \to \mathbb{Z}^{t \times t}$, and non-permutation matrices — with additional considerations regarding security and on the noise growth induced by γ. ([31]).

$$\mathbf{A}_1 \prod_{i\in[L]} \mathbf{D}_{i,x_{\iota(i)}} \approx \left(\prod_{i\in[L]} \mathbf{P}_{i,x_{\iota(i)}} \otimes \mathbf{S}_{i,x_{\iota(i)}}\right) \cdot \mathbf{A}_{L+1} \tag{6.2.5}$$

$$= \prod_{i=1}^{L} \mathbf{P}_{i,x_{\iota(i)}} \otimes \left(\prod_{i=1}^{L} \mathbf{S}_{i,x_{\iota(i)}}\right) \cdot \mathbf{A}_{L+1} \tag{6.2.6}$$

$$= \hat{\mathbf{P}} \otimes \left(\prod_{i=1}^{L} \mathbf{S}_{i,x_i}\right) \cdot \mathbf{A}_{L+1} \tag{6.2.7}$$

Finally, by Theorem 6.2.1 we have that $\hat{\mathbf{P}} \in \{\mathbf{I}, \mathbf{P}^*\}$ corresponding to the bit $C(x)$.

□

While encoding permutation matrices into LWE samples was first described in [34], the first security proof of any GGH15 style construction (regardless of permutation matrices) was shown by Brakerski et al. [26] and by Canetti and Chen in [27]. In particular, the basic semantic security property we wish to accomplish is that the encoding hides the branching program. Canetti and Chen showed this through a "trapdoor closing" strategy, and a more expressive analysis of the LWE assumption which allows for limited additional structure, i.e., the exact same construction written above where we take *structured* secrets $\mathbf{P}_{i,b} \otimes \mathbf{S}_{i,b}$. Based on the definition of LWE, security of such a variant doesn't readily follow. Canetti and Chen showed that this is still secure by reducing LWE to this form, incurring a small $\mathsf{poly}(w)$ blowup in the dimensions.

When considering security properties of GGH15 encodings, it is relevant to consider that the correctness of evaluation *does not* depend on any of the intermediate vertices $\mathbf{A}_i$ being public, but in order to invoke a standard LWE assumption: $(\mathbf{A}, \mathbf{AS} + \mathbf{E}) \approx_c (\mathbf{A}, \mathbf{U})$ at least one component of the tuple must be public. The proof strategies in [27] and [58] assume that all matrices $\mathbf{A}_i$ are global public parameters. Of course, the LWE assumption was originally stated with the term $\mathbf{A}$ as being public, and this is not harmful for security. However, we can instead treat them as auxiliary input in a simulation security definition ([31]). This interpretation will be crucial to our construction.

Security analysis for GGH15 was made more flexible in [31] by defining security with respect to some fixed auxiliary input. They also showed that it is possible to invert the role of the $\mathbf{S}_{i,b}$ and $\mathbf{A}_{i,b}$ terms and treat the low-norm $\mathbf{S}_{i,b}$ matrices as the public keys (and as auxiliary input in the proof), and rely on a modified LWE assumption. This

change allows for improved efficiency and extended functionality in some cases. Informally, semantic security will state that for random-looking $\mathbf{S}_{i,b}, \mathbf{A}_i$ and appropriate auxiliary input, a GGH15 encoding hides the functionality of a branching program. We now formally state the semantic security property, and restate a proof similar to those in [27], [58], [31].

Theorem 6.2.3 (GGH15 Semantic Security ([27], [42], [58], [31]))**.** *Let $L \in \mathbb{N}$. Assuming $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\sigma},U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma^*}}$, for all permutation matrices $\{\mathbf{P}_{i,b} \leftarrow \{0,1\}^{w\times w}\}_{i\in[L],b\in\{0,1\}}$, we have*

$$(\mathsf{aux}, \{\mathbf{D}_{i,b}\}_{i\in[L],b\in\{0,1\}}) \approx_c (\mathsf{aux}, \{\widetilde{\mathbf{D}}_{i,b} \xleftarrow{\$} D_{\mathbb{Z},\sigma}^{m\times m}\}_{i\in[L],b\in\{0,1\}}),$$

where

$$\begin{gathered}\mathbf{A}_{L+1} \xleftarrow{\$} \mathbb{Z}^{t\times m}\\ \{(\mathbf{A}_i, \tau_i) \leftarrow \mathsf{TrapGen}(1^t, 1^m, q)\}_{i\in[L]}\\ \{(\mathbf{S}_{i,b}, \mathbf{E}_{i,b}) \leftarrow D_{\mathbb{Z},\sigma}^{n\times n} \times \chi^{t\times m}\}_{i\in[L],b\in\{0,1\}}\\ \mathbf{B}_{i,b} = (\mathbf{I}_w \otimes \mathbf{S}_{i,b}) \cdot (\mathbf{P}_{i,b} \otimes \mathbf{I}_n \cdot \mathbf{A}_{i+1}) + \mathbf{E}_{i,b}\\ \mathbf{D}_{i,b} \leftarrow \mathbf{A}_i^{-1}(\mathbf{B}_{i,b}, \sigma),\end{gathered}$$

and auxiliary input $\mathsf{aux} = \{\mathbf{A}_i\}_{i\in[L]}$.

Proof. The proof follows the framework of [27], [42], and [58]. In particular, for starting at the end of the "chain", for $i \in [L, 1]$ we will first replace the LWE samples $\mathbf{B}_{i,b}$ with uniformly random, and then rely on Lemma 5.3.2 to argue that preimages of uniformly random matrices are statistically close to $D_{\mathbb{Z},\sigma}^{m\times m}$, and hence can be sampled without requiring the trapdoors.

H_{L+1}**:** This is the real setting. For all i, b: $\mathbf{D}_{i,b} \leftarrow \mathsf{PreSample}(\mathbf{T}_{\mathbf{A}_i}, \mathbf{B}_{i,b})$.

For the next two hybrids we iterate over $0, 1$, and then $b \in \{0, 1\}$ before proceeding to $\mathsf{H}_{i,2}$.

$\mathsf{H}_{i,b,0}$**:** $\mathbf{B}_{i,b}$ is replaced by $\mathbf{U}_{i,b} \xleftarrow{\$} \mathbb{Z}_q^{t\times m}$ for all $j \in [L, i]$, the preimage becomes $\mathbf{D}_{j,b} \leftarrow \mathbf{A}_i^{-1}(\mathbf{U}_{j,b}, \sigma)$.

$\mathsf{H}_{i,b,1}$**:** The preimage is sampled obliviously for all $j \in [L, i] : \widetilde{\mathbf{D}}_{j,b} \xleftarrow{\$} D_{\mathbb{Z},\sigma}^{m\times m}$

$\mathsf{H}_{i,2}$**:** $\mathbf{A}_{j+1} \xleftarrow{\$} \mathbb{Z}_q^{t\times m}$ (rather than being sampled with a trapdoor)

H_0**:** This is the simulated setting. For all i, b: $\{\widetilde{\mathbf{D}}_{i,b} \xleftarrow{\$} D_{\mathbb{Z},\sigma}^{m\times m}\}$, $\mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_q^{t\times m}$.

If there exists an adversary $\mathcal{A}$ whose change in advantage between $\mathsf{H}_{i,b,0}$ and $\mathsf{H}_{i-1,b,0}$ is some non-negligible ϵ, we can create an adversary $\mathcal{B}$ with the same advantage of breaking $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\sigma},U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma^*}}$. The intuition is as follows:

$\mathcal{B}$ will replicate the process of encoding a branching program, starting at the beginning of $\mathsf{BP}(C)$, and until some fixed step $i \in [L]$. In the real setting, i exists outside of this range. During each set of hybrids, we will encode progressively decreasing portions of $\mathsf{BP}(C)$. At a particular step i, for a permutation matrix $\mathbf{P}_{i,b}$ will issue a challenge to an LWE challenger to obtain the correct sample, and then encode a preimage for whatever the challenger returns.

If they return a uniform sample, then the preimage can be generated independently of the trapdoor as it is statistically close to an arbitrary sample in $D_{\mathbb{Z},\sigma}^{m\times m}$ by Lemma 5.3.2. At this point, any trapdoors, and LWE samples for a step $i' > i$ no longer need to be generated. By the previous observation, they are removed from the view of $\mathcal{A}$ because the preimages which would normally induce them are generated independently of the rest of the components. We proceed this way until all the preimages are sampled from $D_{\mathbb{Z},\sigma}^{m\times m}$, and the only remaining matrix $\mathbf{A}_1$ is generated uniformly at random (without a trapdoor).

$\mathcal{B}$ begins by sampling:

$$\{\mathbf{A}_j \leftarrow \mathbb{Z}_q^{t\times m}\}_{j\in[i+1,L+1]} \tag{6.2.8}$$

$$\{(\mathbf{A}_j, \mathbf{T}_{\mathbf{A}_j}) \leftarrow \mathsf{TrapGen}(1^t, 1^m, q)\}_{j\in[1,L-i+1]} \tag{6.2.9}$$

$$\{(\mathbf{S}_{j,b}, \mathbf{E}_{j,b}) \leftarrow D_{\mathbb{Z},\sigma}^{n\times n} \times \chi^{t\times m}\}_{j\in[1,i-1],b\in\{0,1\}} \tag{6.2.10}$$

Then, for each $j \in [L, i]$ and $b \in \{0, 1\}$, it creates $\mathbf{B}_{j,b} = (\mathbf{I}_w \otimes \mathbf{S}_{j,b}) \cdot (\mathbf{P}_{j,b} \otimes \mathbf{I}_n) \cdot \mathbf{A}_{j+1} + \mathbf{E}_{j,b}$, and samples $\mathbf{D}_{j,b} \leftarrow \mathbf{A}_i^{-1}(\mathbf{B}_{j,b}, \sigma)$.

When reaching step i, fix $b = 0$. Instead of generating $\mathbf{B}_{i,b}^*$ itself, it queries the LWE challenger with $\mathbf{A}_{i+1}$ and gets back the challenge output:

$$\mathbf{B}_{i,b}^* = \mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{t\times m} \tag{6.2.11}$$

or

$$\mathbf{B}_{i,b}^* = (\mathbf{I}_w \otimes \mathbf{S}_{i,b}^*) \cdot (\mathbf{I}_{nw} \cdot \mathbf{A}_{i+1}) + \mathbf{E}_{i,b}^* \tag{6.2.12}$$

Namely, $(\mathbf{A}_{i+1}, \mathbf{B}_{i,b}^*)$ is an instance of $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\sigma},U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma^*}}$. To obtain a sample relative to the correct form, $\boldsymbol{A}$ right-multiplies $\mathbf{B}_{i,b}^*$ by $(\mathbf{P}_{i,b} \otimes \mathbf{I}_n)$. This block-permutation is an independent reversible operation, so $\mathbf{B}_{i,b}^*$ remains uniform, or it corresponds to LWE sample

of the form required for correctness. Each $n \times m$ block of $\mathbf{E}^*_{i,j}$ is independent, so this process does yield a valid LWE distribution. Note that $\mathcal{A}$ will be able to produce $(\mathbf{A}_{i+1}, \mathbf{B}^*_{i,b})$ since $(\mathbf{A}_{i+1}, \mathbf{D}^*_{i,b})$ are in their view.

Next, $\mathcal{B}$ samples a preimage for the challenge $\mathbf{D}^*_{i,b} : \mathbf{A}_i \mathbf{D}^*_{i,b} = \mathbf{B}^*_{i,b}$, and sends the resulting distribution:

$$(\mathbf{A}_1, \{\mathbf{D}_{1,0}, \ldots, \mathbf{D}^*_{i,b}, \mathbf{D}_{i,1-b}, \mathbf{D}_{i+1,0}, \ldots \mathbf{D}_{L,1}\})$$

to $\mathcal{A}$. Now, $\mathcal{A}$'s goal is to distinguish $\mathsf{H}_{i,0,0}$ from $\mathsf{H}_{i-1,2}$. The only thing that changes is the distribution of $\mathbf{D}^*_{i,b}$ conditioned on the distribution of $\mathbf{B}^*_{i,b}$. If $\mathcal{A}$ has any advantage in distinguishing between the two settings, then their advantage translates into $\mathcal{B}$'s advantage against the LWE challenger.

Next, we analyze the difference between $\mathsf{H}_{i,0,0}$ and $\mathsf{H}_{i,0,1}$. By Lemma 5.3.2, the distribution $(\mathbf{A}_j, \mathbf{A}_j^{-1}(\mathbf{U}))$ is statistically close to $(\mathbf{A}_j, \mathbf{U}' \xleftarrow{\$} D^{m \times m}_{\mathbb{Z},\sigma})$. So, we replace $\mathbf{D}^*_{i,b}$ with a uniform discrete Gaussian sample, and the distributions remain statistically close. We repeat this set of changes for $b = 1$ (since $\mathbf{S}_{i,b}$ is sampled independently), resulting in replace both $\mathbf{D}_{j,b}$ being replaced with uniform discrete Gaussian samples $\widetilde{\mathbf{D}}_{j,b}$ for $j > i$.

Then, observe that no LWE samples are ever produced under $\mathbf{A}_{j+1}$. So, we "close" the trapdoor by replacing it with a uniformly random matrix, that is: $\mathbf{A}_{j+1} \xleftarrow{\$} \mathbb{Z}_q^{t \times m}$. Since $\mathbf{A}_{j+1}$ is statistically close to uniform over $\mathbb{Z}_q^{t \times m}$ the change between $\mathsf{H}_{i,1,1}$ and $\mathsf{H}_{i,2}$ is also statistically close. This completes one iteration of changes for an index i.

Finally, after iterating over all $i \in [L, 1]$, after making the change $\mathsf{H}_{1,1,1}$ we also replace $\mathbf{A}_1$ with uniformly random, as we no longer need to use its trapdoor. As above, this change is statistically close, and moreover none of the $\{\widetilde{\mathbf{D}}_{i,b}\}$ depend on $\mathbf{A}_1$ since they are sampled obliviously. Now, we are in H_0 where the entire distribution is simulated.

□

6.3 Construction

6.3.1 Construction Overview

Now, we describe how to use the GGH15 encoding strategy to instantiate Privately Constrained Testable PRFs for a constraint family $\mathcal{C}_\lambda$ in NC^1 which are testable for both ranges. For the remainder of this section let $w = 5$, and let $\boldsymbol{e}_i$ be the i-th standard basis vector of a w-dimensional vector space.

Following [31]), we define the notation: for $k \in [w]$ let $\overline{\mathbf{A}}^{(k)}$ denote the $(k-1)n+1$ to kn-th rows of a matrix $\mathbf{A} \in \mathbb{Z}^{wn \times m}$ (i.e. $(\boldsymbol{e}_k^T \otimes \mathbf{I}_n) \cdot \mathbf{A}$), and extend the definition to ranges of indices $\overline{\mathbf{A}}^{[i,j]}$. Additionally, for all ensembles $(x \in \{0,1\}^L, \{\mathbf{V}_{i,b}\}_{i \in [L], b \in \{0,1\}})$ define: $\mathbf{V}_x = \prod_{i \in [L]} \mathbf{V}_{i,x_i}$. We take all terms indexed by x_i as their unique representative $x_{\iota(i)}$, with the input to index map omitted for brevity.

Testability. Our starting point is Eq. (6.1.1) due to Bannerjee et al. [10].

$$\mathsf{F}_{\mathbf{A},\{\mathbf{S}_{i,b}\}}(x) = \lfloor \mathbf{S}_x \cdot \mathbf{A} \rceil_p$$

The first observation is that by sampling $\mathbf{A}$ together with a trapdoor, it is easy to sample a *low-normed* $\mathbf{Z} : \mathbf{AZ} \equiv \mathbf{0} \bmod q$. Then, for any any PRF output $\mathbf{Y}$, $\mathbf{YZ} \approx \mathbf{0}$ up to some rounding error. For a random matrix $\mathbf{Y}'$ (outside of the range of F whp.), the resulting product will have a sufficiently small ℓ_∞ norm with only small probability. This forms the basis of the test. Using a "special" $\mathbf{A}$ rather than uniform does not contradict any security definitions since the resulting $\mathbf{A}$ can be made statistically close to uniform.

In the actual construction we will only sample *half* of $\mathbf{A} = [\mathbf{A}' \| \mathbf{R}]$ with a trapdoor to simplify the proof of unique testability. Then, for low-normed $\mathbf{Z} : \mathbf{A}'\mathbf{Z} \equiv \mathbf{R} \bmod q$ — we will parse $\mathbf{Y} = [\mathbf{Y}_0 \| \mathbf{Y}_1]$ and check if $\mathbf{Y}_0\mathbf{Z} - \mathbf{Y}_1 \approx 0$.

Two Ranges. Since the testability strategy only makes sense for a single-ranged PRF, we proceed with the obvious strategy of simply sampling two different matrices $\mathbf{A}_0^*$, and $\mathbf{A}_1^*$ structured as above with separate trapdoors, giving us two separate PRFs. Then, all we need to do is have a way of embedding the two PRF keys $\mathbf{A}_0^*, \mathbf{A}_1^*$ into some kind of gadget which outputs PRF evaluations under $\mathbf{A}_b^*$ for $b = C(x)$, while hiding the keys and C. We can accomplish this using GGH15 encodings.

Assembling the Pieces. Now, we can use a GGH15 encoding of a circuit C. Lemma 6.2.1 tells us that:

$$\mathbf{A}_1 \cdot \mathbf{D}_x \approx (\mathbf{I}_n \otimes \mathbf{S}_x) \cdot (\mathbf{P}_b \otimes \mathbf{I}_n \cdot \mathbf{A}_{L+1})$$

For, $\mathbf{P}_b \in \{\mathbf{P}^*, \mathbf{I}_w\}$ a fixed permutation which depends only on $b = C(x)$ (cf. Definition 6.2.1). So, the permutation moves a fixed distinct $n \times m$ block into the first n rows of $\mathbf{A}_{L+1}$. So, we choose $\mathbf{A}_{L+1}$ such that it contains $\mathbf{A}_0^*$ and $\mathbf{A}_1^*$, and in particular so that $\mathbf{P}_b$ moves $\mathbf{A}_b^*$ into the first n rows of $\mathbf{A}_{L+1}$ during evaluation.

Then, by publishing only the top n rows of $\mathbf{A}_1$ (i.e. $\overline{\mathbf{A}}_1^{(1)}$) we guarantee the output of the GGH15 evaluation is an $n \times m$ matrix in the image of $\mathbf{A}_b^*$. Then, rounding the output gives us a PRF.

$$\left\lfloor \overline{\mathbf{A}}_1^{(1)} \cdot \mathbf{D}_x \right\rceil_p = \left\lfloor \mathbf{S}_x \cdot \mathbf{A}_b^* \right\rceil_p$$

Hiding the bottom $(w-1) \cdot n$ rows of also $\mathbf{A}_1$ gives the lossy map needed for security — otherwise an adversary always learns both image points of the PRF wrt. x, and hence always answer challenges correctly in the residual pseudorandomness experiment. Note that our "two PRFs" are not actually independent since the GGH15 encoding reuses the $\{\mathbf{S}_{i,b}\}$ terms, implying their private keys are correlated. We will show both output ranges are pseudorandom by viewing $\{\mathbf{D}_{i,b}\}$ as the global public parameters, and the unpublished blocks of $\mathbf{A}_1$ as the secret key of the Boneh et al. ([16]) PRF in Eq. (6.1.3) instead.

Construction 6.3.1 (Range-testable Privately Constrained PRF)**.**

$\mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$**:**

1. *Take* $L = 4^d$ *for* $d = \log(\ell_{in})$
2. *Choose* n, q, σ*, set* $t = wn$*,* $m = \Omega(t \log q)$ *as per Section 6.3.2*
3. *Sample* $(\mathbf{A}'_0, \mathbf{T}_{\mathbf{A}'_0}), (\mathbf{A}'_1, \mathbf{T}_{\mathbf{A}'_1}) \leftarrow \mathsf{TrapGen}(1^n, 1^{(m/2)}, q)$
4. *Sample* $\mathbf{R}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times m/2}, \mathbf{R}_1 \xleftarrow{\$} \mathbb{Z}_q^{n \times m/2}$
5. *Let* $\mathbf{A}^*_0 = [\mathbf{A}'_0 \parallel \mathbf{R}_0], \mathbf{A}^*_1 = [\mathbf{A}'_1 \parallel \mathbf{R}_1]$
6. *Sample* $\mathbf{Z}_0 \leftarrow \mathsf{PreSample}(\mathbf{T}_{\mathbf{A}'_0}, \mathbf{R}_0, \sigma), \mathbf{Z}_1 \leftarrow \mathsf{PreSample}(\mathbf{T}_{\mathbf{A}'_1}, \mathbf{R}_1, \sigma)$
7. *Sample* $\mathbf{A}_{L+1} \xleftarrow{\$} \mathbb{Z}_q^{wn \times m}$.
8. *Sample* $\{\mathbf{S}_{i,b} \leftarrow D^{n \times n}_{\mathbb{Z},\sigma}\}_{i \in [L], b \in \{0,1\}}$
9. *Set*

$$\overline{\mathbf{A}}^{[1,2]}_{L+1} = \begin{bmatrix} \mathbf{A}^*_1 \\ \mathbf{A}^*_0 \end{bmatrix}$$

10. *Output* $\mathsf{sk} = (\{\mathbf{S}_{i,b}\}, \mathbf{A}^*), \mathsf{pp} = (p, q, t, m)$

$\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b)$**:**

1. *Compute* $\mathbf{S}_x = \prod_{i \in [L]} \mathbf{S}_{i,x_i}$
2. *Output* $y = \left\lfloor \mathbf{S}_x \cdot \mathbf{A}^*_b \right\rceil_p$

$\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$**:**

1. *Parse* C *as a length* L*, width* w *branching program:* $\{\mathbf{P}_{i,b}\}_{i \in [L], b \in \{0,1\}}$
2. *Let* $\pi_{i,b} = \mathbf{P}_{i,b} \otimes \mathbf{I}_n$.
3. *Sample* $\{(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}_i}) \leftarrow \mathsf{TrapGen}(1^t, 1^m, q), \mathbf{E}_{i,b} \leftarrow \chi^{t \times m}\}_{i \in [L], b \in \{0,1\}}$
4. *Set* $\mathbf{A}_{L+1} = \mathbf{A}^*$
5. *Compute* $\mathbf{B}_{i,b} = (\mathbf{I}_w \otimes \mathbf{S}_{i,b}) \cdot (\pi_{i,b} \cdot \mathbf{A}_{i+1}) + \mathbf{E}_{i,b}$

6. *Use the trapdoors* $\mathbf{T}_{\mathbf{A}_i}$ *to create the directed encodings of the edges* $A_i \to (\pi_{i,b} \cdot A_{i+1})$*:*

$$(\mathbf{D}_{i,0}, \mathbf{D}_{i,1}) \leftarrow \mathsf{PreSample}(\mathbf{T}_{\mathbf{A}_i}, \mathbf{B}_{i,b}) \mid \mathbf{A}_i \mathbf{D}_{i,b} = \mathbf{B}_{i,b}$$

7. *Compute* $\overline{\mathbf{A}}_1^{(1)} = (\boldsymbol{e}_1^T \otimes \mathbf{I}_n) \cdot \mathbf{A}_1$
8. *Output* $\mathsf{sk}[C] = (\overline{\mathbf{A}}_1^{(1)}, \{\mathbf{D}_{i,b}\}_{i \in [L], b \in \{0,1\}})$

$\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x)$**:**

1. *Parse* $\mathsf{sk}[C] = (\overline{\mathbf{A}}_1^{(1)}, \{\mathbf{D}_{i,b}\}_{i \in [L], b \in \{0,1\}})$
2. *Compute* $\mathbf{D}_x = \prod_{i \in [L]} \mathbf{D}_{i,x_i}$
3. *Output* $y = \left\lfloor \overline{\mathbf{A}}_1^{(1)} \cdot \mathbf{D}_x \right\rceil_p$

$\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y)$**:**

1. *Parse* $y = [\mathbf{Y}_0 \parallel \mathbf{Y}_1]$
2. *Compute* $\mathbf{C} = \mathbf{Y}_0 \cdot \mathbf{Z}_b - \mathbf{Y}_1$
3. *Output* 1 *if* $\|\mathbf{C}\| < q/4$*, otherwise output* 0

6.3.2 Correctness and Parameters

One important difference in our construction from [27], [31] is that we will choose p not simply arbitrarily close to 2 (to output at least one bit out of each position of the resulting product) so that q/p is larger than the error bound introduced during evaluation. Instead, we take p to be large enough so that the absolute error introduced by rounding is still small enough to allow for the correctness of a subspace test (i.e. multiplying by a short preimage). In order to cover all of the correctness and security requirements we will choose parameters as follows:

1. Take $n = O(\lambda)$ for security, $t = wn$ to support encoding $w \times w$ permutation matrices
2. Take $m = \Omega(t \log q)$ to guarantee solutions for any coset with high probability (Lemma 5.2.1), with some hidden constant $c > 4$ to support partitioned testing
3. Set $\chi = D_{\mathbb{Z}, 2\sqrt{n}}$ for hardness of LWE (Definition 5.2.4)

4. Take $\sigma \geq \sqrt{t \log q}$ for satisfiability of trapdoor sampling (Lemma 5.3.1)

The combination of parameters yields $\beta \leq O(t\sqrt{\log q})$ (cf. Remark 5.3.1). As in other works based on GGH15, since the magnitude of the initial LWE error terms needs to be small for correctness, we end up relying on LWE with subexponential approximation factors to the underlying lattice problems.Finally, note that the requirement that $q/p > \beta(2m\beta)^L$ is stronger than the requirement $q/p > \beta^L$ of Lemma 6.3.1, so taking $p > 4m\beta$ is without loss of generality.

Theorem 6.3.1 (Preservation of Functionality for Constrained Inputs)**.** *For all* $x \in \{0,1\}^{\ell_{in}}$ *and* $C \in \mathcal{C}_\lambda$*,*

$$\Pr[\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x) \neq \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, C(x), x)] = \mathsf{negl}(\lambda) \tag{6.3.1}$$

Proof. Consider the expression below:

$$\begin{aligned}
\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x) &= \left\lfloor \overline{\mathbf{A}}_1^{(1)} \cdot \mathbf{D}_x \right\rceil_p && (6.3.2)\\
&= \left\lfloor \boldsymbol{e}_1^T \otimes \mathbf{I}_n \cdot (\mathbf{A}_1 \cdot \prod_{i\in[L]} \mathbf{D}_{i,x_i}) \right\rceil_p && (6.3.3)\\
&= \boldsymbol{e}_1^T \otimes \mathbf{I}_n \left\lfloor (\mathbf{A}_1 \cdot \prod_{i\in[L]} \mathbf{D}_{i,x_\iota}) \right\rceil_p && (6.3.4)\\
&= \boldsymbol{e}_1^T \otimes \mathbf{I}_n \left\lfloor \left(\prod_{i=1}^{L} \hat{\mathbf{S}}_{i,x_i}\right) (\mathbf{P} \otimes \mathbf{I}_n) \cdot \mathbf{A}_{L+1} \right\rceil_p && (6.3.5)\\
&= \boldsymbol{e}_1^T \otimes \mathbf{I}_n \left\lfloor \left(\mathbf{I}_w \otimes \prod_{i=1}^{L} \mathbf{S}_{i,x_i}\right) (\mathbf{P} \otimes \mathbf{I}_n) \cdot \mathbf{A}_{L+1} \right\rceil_p && (6.3.6)\\
&= \left\lfloor \left(\prod_{i=1}^{L} \mathbf{S}_{i,x_i}\right) \cdot \mathbf{A}^*_{C(x)} \right\rceil_p && (6.3.7)\\
&= \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, C(x), x), && (6.3.8)
\end{aligned}$$

which follows from Eq. (6.2.3), and the fact that the bookend has no effect on the norm. For the final step, it is true that for $C(x) = 0$, the matrix $\mathbf{A}_0^*$ does not necessarily end up in the first block of $\mathbf{A}_{L+1}$ for the permutation $\mathbf{P}^*$ as defined in the setup algorithm (as

opposed to the case $C(x) = 1$, where it is the identity matrix). However, since $\mathbf{P}^*$ is some fixed 5-cycle which depends only on L, this is without loss of generality, as we can simply insert $\mathbf{A}_0^*$ into the appropriate block of $\mathbf{A}_{L+1}$ when computing the constrained key.

□

Theorem 6.3.2 (Testing Completeness and Unique Testability)**.** *The algorithm* $\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y)$ *in Construction 6.3.1 satisfies completeness and unique testability.*

Proof. For any input x, consider that,

$$y = \lfloor \mathbf{S}_x \mathbf{A}_b^* \rceil_p \pmod p \equiv \mathbf{S}_x \mathbf{A}_b^* + \hat{\mathbf{E}} \pmod q, \tag{6.3.9}$$

where $\hat{\mathbf{E}}$ is the absolute error introduced by rounding, e.g. $\|\hat{\mathbf{E}}\|_\infty = O(\beta(2m\beta)^{L-1})$.

First, parse $\mathbf{A}_b^* = [\mathbf{A}'_b \parallel \mathbf{R}_b]$, and $y = [\mathbf{Y}_0 \parallel \mathbf{Y}_1]$. Then,

$$[\mathbf{Y}_0 \parallel \mathbf{Y}_1] = [\mathbf{S}_x \mathbf{A}'_b + \hat{\mathbf{E}}_0 \parallel \mathbf{S}_x \mathbf{R}_b + \hat{\mathbf{E}}_1] \tag{6.3.10}$$

For $\mathbf{Z}_b \in \chi^{m \times m} : \|\mathbf{Z}_b\|_\infty \leq \beta \wedge \mathbf{A}'_b \mathbf{Z}_b \equiv \mathbf{R}_b \pmod q$:

$$(\mathbf{S}_x \mathbf{A}'_b + \hat{\mathbf{E}}_0) \cdot \mathbf{Z}_b = \mathbf{S}_x \mathbf{R}_b + \hat{\mathbf{E}}_0 \cdot \mathbf{Z}_b \tag{6.3.11}$$

$$\approx \mathbf{Y}_1, \tag{6.3.12}$$

where we bound the error as $\hat{\mathbf{E}}_1 + \|\hat{\mathbf{E}}_0 \cdot \mathbf{Z}_b\|_\infty \leq 2 \cdot \|\hat{\mathbf{E}}_0 \cdot \mathbf{Z}_b\|_\infty \leq m\beta q/p$. So, testing is complete for $p > 4m\beta$ as we can unambiguously check the distance modulo q from $\mathbf{R}_b$.

Next, observe that by Lemma 5.3.2, since each $\mathbf{Z}_b$ is a preimage of a uniformly random matrix, if $[\mathbf{Y}_0 \| \mathbf{Y}_1] \in \mathcal{R}_b$, then $\mathbf{Y}_0 \cdot \mathbf{Z}_{1-b}$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$. ($\mathbf{A}'_b$ is statistically close to uniform, and therefore generates $\mathbb{Z}_q^n$ with probability exponentially close to 1). So, the expression $(\mathbf{Y}_0 \cdot \mathbf{Z}_{b-1} - \mathbf{Y}_1) \bmod q$ is itself close to uniform, and it follows that:

$$\Pr[\|(\mathbf{Y}_0 \cdot \mathbf{Z}_{b-1} - \mathbf{Y}_1) \bmod q\|_\infty < q/4] = 2^{-nm} = \mathsf{negl}(\lambda)$$

Hence, the probability that both tests pass is negligible in the security parameter, as required.

□

6.3.3 Security

Security relies on the lower $n(w-1)$ rows of $\mathbf{A}_1$ being hidden from the adversary. This follows from the proof of Theorem 6.2.3 since the preimages themselves are computationally close to random discrete gaussian samples, and we never need to produce any LWE samples *under* $\mathbf{A}_1$. Although the *entirety* of $\mathbf{A}_1$ is considered part of the auxiliary input available to the adversary and simulator in the GGH15 security experiment; this does not contradict our requirements for the PRF. Since the GGH15 adversary is participating in a distinct experiment, and produces constrained keys themselves — they do not gain any inherent advantage from seeing these values and can simulate the PCT-PRF challenger appropriately.

A Weaker Security Definition. Unfortunately, it is not clear how to show that our construction satisfies the strong definition of Definition 3.1.1. In particular, transitioning between encoding a circuit C_0 to a circuit C_1 in a series of hybrids (to preserve testing oracle access) is a challenge without a clear resolution. This challenge occurs specifically because of limitations in the current understanding of GGH15 encodings. To this end, we introduce a weaker simulation-based definition which ignores the testability properties altogether (since it is *also* unclear how to reconcile testing oracles with simulated constrained keys). This security definition restricts attention to just the "two-sided" residual pseudorandomness and constraint privacy. The definition is otherwise similar to the Definition 2.2.4 which is common to all recent works on Privately Constrained (non-testable) PRFs.

In applications, this means testing correctness must be lost during certain stages of the security proof when changing between constraint circuits. This poses a serious challenge to successfully proving security of applications relative to this definition, and we are presently unable to do so. Furthermore, the construction of PCT-PRFs from GGH15 may not satisfy conditional one-wayness of Definition 3.1.1, we explore this in Section 7.1.

Definition 6.3.1 (Range-testable Privately Constrained PRF (Weak) Simulation Security)**.** *For any stateful* PPT *algorithm* $\mathcal{A}$*, there exists a* PPT *stateful Simulator* $\mathcal{S}$ *such that:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathcal{S}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

$\mathcal{A}$ may ask a single constraint query for some circuit $C \in \mathcal{C}_\lambda$ followed by a polynomially bounded number of evaluation queries (x, b_x). In the ideal experiment, the constrained key is produced by a simulator which receives only the size of the constraint ℓ_C. In the ideal experiment, the simulator also learns an indicator bit $d_x = C(x)$ to enforce consistency. If $d_x \neq b_x$, $\mathcal{S}$ answers evaluation queries by (statefully) sampling a uniformly random value from the co-domain. Otherwise, it answers by evaluating using the simulated key. The output of the experiment is the output bit of $\mathcal{A}$.

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)}$*:*

1: $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\ell_C}, 1^{\ell_{in}})$
2: $\mathcal{A} \rightarrow C$
3: $\mathsf{sk}[C] = \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}, C)$
4: $\mathcal{A} \leftarrow \mathsf{sk}[C]$
5: *Repeat:*
6: $\quad \mathcal{A} \rightarrow (x, b_x)$
7: $\quad y = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}, x, b_x)$
8: $\quad \mathcal{A} \leftarrow y$
9: $\mathcal{A} \rightarrow b$; *Output* b

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathcal{S}}(1^\lambda)}$*:*

1: $\mathcal{S} \leftarrow 1^\lambda$
2: $\mathcal{A} \rightarrow C$
3: $\mathsf{sk}[\widetilde{C}] = \mathcal{S}(1^{\ell_C})$
4: $\mathcal{A} \leftarrow \mathsf{sk}[\widetilde{C}]$
5: *Repeat:*
6: $\quad \mathcal{A} \rightarrow (x, b_x)$; $d_x = C(x)$
7: $\quad$ ***if*** $d_x = b_x$ ***then***
8: $\quad\quad y' = \mathcal{S}(x, b_x, d_x)$
9: $\quad$ ***else***
10: $\quad\quad y' \overset{\$}{\leftarrow} \mathcal{Y}$
11: $\quad$ ***end if***
12: $\quad \mathcal{A} \leftarrow y'$
13: $\mathcal{A} \rightarrow b$; *Output* b

Lemma 6.3.1 (BLMR PRF (Adapted from [27], [16]))**.** *Let $L \in \mathbb{N}$ be the bit-length of an input x, $n, m, p, q, \in \mathbb{N}$. Let $\sigma, \sigma^* \in \mathbb{R} : 0 < \sigma < \sigma^* < q$, and $q/p > (\sigma^*\sqrt{m})^L$. Let $\upsilon_q = U(\mathbb{Z}_q^{n\times m}), \gamma_\sigma = D_{\mathbb{Z},\sigma}^{m\times m}$, and $\chi_{\sigma^*} = D_{\mathbb{Z},\sigma^*}^{n\times m}$. For any $\mathbf{U} \leftarrow \upsilon_q$, and $\{\mathbf{D}_{i,b} \leftarrow \gamma_\sigma\}_{i\in[L],\{0,1\}}$, the function $\mathsf{F} : \{0,1\}^L \to \mathbb{Z}_p^{n\times m}$ defined as:*

$$\mathsf{F}_{\mathbf{U}}(x) = \left\lfloor \mathbf{U} \cdot \prod_{i\in[L]} \mathbf{D}_{i,x_i} \right\rceil_p$$

is a PRF assuming the hardness of $\mathsf{LWE}_{n,m,q,\upsilon_q,\gamma_\sigma,\chi_{\sigma^*}}$.

Theorem 6.3.3 (Simulation Security)**.** *Construction 6.3.1 satisfies (weak) simulation security as per Definition 6.3.1 assuming $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\sigma},U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma}}$ and $\mathsf{LWE}_{n,m,q,\upsilon_q,\gamma_\sigma,\chi_{\sigma^*}}$ as per Lemma 6.3.1.*

Proof. We define the simulated constrained key sampling procedure for $\mathcal{S}(1^\lambda)$ as follows:

1. Upon receiving a constrained key query for $C \in \mathcal{C}_\lambda$. Let $L = \mathsf{poly}(\ell_C, \ell_{in})$, the simulator $\mathcal{S}$ samples $\mathsf{sk}[\widetilde{C}] = (\mathbf{U} \leftarrow \mathbb{Z}_q^{n\times m}, \{\mathbf{D}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{m\times m}\}_{i\in[L],b\in\{0,1\}})$, and outputs $\mathsf{sk}[\tilde{C}]$ as the constrained key.
2. For any evaluation query (x, b_x) and indicator bit d_x, $\mathcal{S}$ outputs:

$$\mathsf{P.Eval}_{\mathcal{S}}(x) = \begin{cases} \lfloor \mathbf{U} \cdot \mathbf{D}_x \rceil_p & \text{if } d_x = b_x \\ \mathbf{Y} \xleftarrow{\$} \mathbb{Z}_p^{n\times m} & \text{otherwise} \end{cases}$$

We proceed in two steps, first we define a simulator $\mathcal{S}^*$ and prove its behaviour is indistinguishable from the real experiment assuming semantic security of GGH15 encodings for permutation branching programs (Theorem 6.2.3). Then, we use Lemma 6.3.1 to show that $\mathcal{S}^*$ is computationally indistinguishable from $\mathcal{S}$.

Before describing the behaviour of $\mathcal{S}^*$, we describe how given a constrained key in the form of a GGH15 branching program encoding $((\boldsymbol{e}_1^T \otimes \mathbf{I}_n) \cdot \mathbf{A}_1, \{\mathbf{D}_{i,b}\})$, for any input x the image of $(x, 1 - C(x))$ can always be recovered given the entirety of $\mathbf{A}_1$ (without the bookend) since evaluations of this form always produce *both* possible images under x. Moreover, since any length L oblivious branching program induces a fixed w-cycle $\mathbf{P}^*$ — we can output the image of $(x, 1 - C(x))$ directly by "shifting" the phase of the cycle.

Recall that the output of any branching program evaluation is a permutation matrix $\mathbf{P} \in \{\mathbf{I}, \mathbf{P}^*\}$. Let $\bar{\mathbf{P}}$ denote the complement of $\mathbf{P}$ in this set. In particular, $\mathbf{P} = \mathbf{I}$ if $b = C(x) = 1$, and $\bar{\mathbf{I}} = \mathbf{P}^*$. We first observe that for any permutation matrix $\mathbf{P}$, and every $\gamma \in [w]$, there exists $\alpha \in [w]$ such that $\boldsymbol{e}_\alpha^T \cdot \mathbf{P} = \boldsymbol{e}_\gamma^T$. For example, if $\mathbf{P} = \mathbf{I}$ then we have the trivial relation $\alpha = \gamma, \forall \alpha \in [w]$. Then, for any $\bar{\mathbf{P}}$ we condition α on $\boldsymbol{e}_\gamma \cdot \mathbf{P} = \boldsymbol{e}_1 \cdot \bar{\mathbf{P}}$ (corresponding to the permutation which represents $1 - b$):

$$\left\lfloor (\mathbf{S}_x \cdot \mathbf{A}^*_{1-b}) \right\rceil_p = \left\lfloor (\boldsymbol{e}_1^T \otimes \mathbf{I}_n) \cdot (\mathbf{I}_w \otimes \mathbf{S}_x) \cdot (\boldsymbol{e}_\alpha^T \mathbf{P} \otimes \mathbf{I}_n) \cdot \mathbf{A}_{L+1} \right\rceil_p \tag{6.3.13}$$

$$= \left\lfloor (\boldsymbol{e}_\alpha^T \otimes \mathbf{I}_n) \cdot \mathbf{A}_1 \cdot \mathbf{D}_x \right\rceil_p, \tag{6.3.14}$$

where equality holds via Theorem 6.3.1 and the mixed-product property of the Kronecker product.

Now, define the functionality of $\mathcal{S}^*$ as follows:

1. Upon receiving a constrained key query for $C \in \mathcal{C}_\lambda$, the simulator $\mathcal{S}^*$ samples $\mathsf{sk}[\widetilde{C}] = (\mathbf{U} \leftarrow \mathbb{Z}_q^{t \times m}, \{\mathbf{D}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{m \times m}\}_{i \in [L], b \in \{0,1\}})$, and outputs $\mathsf{sk}[\widetilde{C}]$ as the constrained key.

2. Let $\overline{\mathbf{U}}^{(i)} = (\boldsymbol{e}_i \otimes \mathbf{I}_n) \cdot \mathbf{U}$. For any evaluation query (x, b_x) and indicator bit d_x, $\mathcal{S}$ outputs:

$$\mathsf{P.Eval}_{\mathcal{S}^*}(x) = \begin{cases} \left\lfloor \overline{\mathbf{U}}^{(1)} \cdot \mathbf{D}_x \right\rceil_p & \text{if } d_x = b_x \\ \left\lfloor \overline{\mathbf{U}}^{(a)} \cdot \mathbf{D}_x \right\rceil_p & \text{else if } b_x = 0 \\ \left\lfloor \overline{\mathbf{U}}^{(b)} \cdot \mathbf{D}_x \right\rceil_p & \text{else if } b_x = 1, \end{cases}$$

where we fix a, b such that

$$\boldsymbol{e}_a = \boldsymbol{e}_\gamma^T \cdot \mathbf{I} = \boldsymbol{e}_1^T \cdot \mathbf{P}^* \tag{6.3.15}$$

$$\boldsymbol{e}_b = \boldsymbol{e}_\gamma^T \cdot \mathbf{P}^* = \boldsymbol{e}_1^T \cdot \mathbf{I} \tag{6.3.16}$$

These indices are uniquely determined by the length of the branching program only, so it is without loss of generality that the simulator can choose these values without knowing C.

Lemma 6.3.2. *The real experiment is indistinguishable from the output of $\mathcal{S}^*$ assuming $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\sigma},U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma}}$.*

Proof. The proof proceeds via semantic security of GGH15 encodings as per Theorem 6.2.3 with auxiliary input $\{\mathbf{A}_i\}$ in a similar fashion to the proof of Lemma 7.6 in [31]. Namely, if there exists an adversary $\mathcal{A}$ which distinguishes the real distribution from the output of $\mathcal{S}^*$, we can create an efficient adversary $\mathcal{B}$ which distinguishes between the distributions in Theorem 6.2.3.

1. $\mathcal{A}$ queries $\mathcal{B}$ for a constrained key for a circuit C. $\mathcal{B}$ queries the GGH15 challenger with $\mathsf{BP}(C) = \{\mathbf{P}_{i,b}\}_{i\in[L],b\in\{0,1\}}$ and receives $\{\mathbf{A}_i\}_{i\in[L]}, \{\mathbf{D}_{i,b}\}_{i\in[L],b\in\{0,1\}}$.

2. $\mathcal{B}$ answers evaluation queries as follows:

$$\mathsf{P.Eval}_{\mathcal{B}}(x) = \begin{cases} \left\lfloor \overline{\mathbf{A}}_1^{(1)} \cdot \mathbf{D}_x \right\rceil_p & \text{if } d_x = b_x \\ \left\lfloor \overline{\mathbf{A}}_1^{(a)} \cdot \mathbf{D}_x \right\rceil_p & \text{else if } b_x = 0 \\ \left\lfloor \overline{\mathbf{A}}_1^{(b)} \cdot \mathbf{D}_x \right\rceil_p & \text{else if } b_x = 1 \end{cases}$$

Since $\mathbf{A}_i$ are chosen statistically close to uniform in the real setting, the first component of the product above is statistically close to the distribution of $\mathbf{U}$ in $\mathcal{S}^*$. Then, if $\{\mathbf{D}_{i,b}\}_{i\in[L],b\in\{0,1\}}$ corresponds to a constrained key $\mathsf{sk}[C]$ we are in the real setting. Otherwise, if the outputs are simulated by the GGH15 challenger, by LWE the output is computationally indistinguishable from the distribution of $\{\mathbf{D}_{i,b}\}_{i\in[L],b\in\{0,1\}}$ produced by $\mathcal{S}^*$. Then, when $\mathcal{A}$ responds with "simulated" or "real", $\mathcal{B}$ forwards the same reply to the GGH15 challenger. Hence, the advantages of $\mathcal{B}$ and $\mathcal{A}$ are the same.

□

Finally, we observe that the distributions of $\mathcal{S}$ and $\mathcal{S}^*$ are computationally close assuming $\mathsf{LWE}_{n,m,q,\upsilon_q,\gamma_\sigma,\chi_{\sigma^*}}$ through a hybrid argument by treating $\overline{\mathbf{A}}_1^{(a)}$ and then $\overline{\mathbf{A}}_1^{(b)}$ as the secret keys in Lemma 6.3.1, which concludes the proof of the theorem.

□

Chapter 7

Extensions and Discussions

7.1 Conditional One-wayness

We can consider standard PRFs to be one-way in a straightforward manner because the secret key is not delegated, and the values themselves are not *testable*. However, despite our construction satisfying most of the desired properties based on LWE, it is not clear how to argue one-wayness given the secret key from the same assumption, or any assumption at all.

Consider an *unrounded* output of the PCT-PRF Construction 6.3.1: $\hat{y} = \mathbf{S}_x\mathbf{A}_b^* + \mathbf{E}_x$. By examining the correctness of evaluation in Theorem 6.2.2, we see that both the term $\mathbf{S}_x$ and $\mathbf{E}_x$ not only depend on the input x, but do not take the form of anything resembling a computational assumption related to lattice problems. So, if we could recover both terms from $\hat{y}$, we could potentially learn something about x. Since $\mathbf{A}_b^*$ is sampled together with a trapdoor, we can use the LWE inversion algorithms of [35] or [50] to do exactly this.

We can deal with leaking $\mathbf{E}_x$ simply by only returning rounded outputs $\lfloor \hat{y} \rceil_p$ (as defined in the construction already), since the absolute error introduced by rounding statistically loses information about the original error term. However, rounding does not impact the inversion algorithm ([6]) as long as the magnitude of the rounding error is bounded by $q/4$ (which we require for correctness of testability.) Hence, the holder of the trapdoor of $\mathbf{A}_b^*$ can always recover $\mathbf{S}_x$ as well. The problem of recovering the input given $\mathbf{S}_x$ is captured by the following definition.[7]

[7]Note, that in our setting the problem is far more structured since for an input-repeating branching program $\ell = L = (\ell_{in})^2$. So, in fact our product takes indices which are periodic over ℓ_{in} with respect to $\iota(i)$.

Definition 7.1.1 (Low-Dimension Discrete Gaussian Subset-Product)**.** *For* $n \in \mathbb{N}, q \leq 2^n$, *and* $\sigma = \mathsf{poly}(n)$, *given* $\{\mathbf{S}_{i,b} \leftarrow D^{n\times n}_{\mathbb{Z},\sigma}\}_{i\in\ell,b\in\{0,1\}}$, *and a target* $\mathbf{R} \in \mathbb{Z}^{n\times n}_q$, *find* $x \in \{0,1\}^\ell$: $\mathbf{R} = \mathbf{S}_x = \prod_{i\in[\ell]} \mathbf{S}_{i,x_i} \bmod q$.

Impagliazzo and Naor showed in [45] that the Subset-Product of n group elements in some group $\mathbb{G}$ is one-way if $\ell > c \log|\mathbb{G}|$ for any $c > 1$, or in particular if the function is compressing. This is at odds with our typical desire for PRFs to be injective, moreover if we treat $\mathbf{S}_{i,b}$ as elements of $\mathbb{G} = GL(n, q)$ for some prime power q then $\log|\mathbb{G}| = n(n-1)\log q$. This is substantially larger than what is attainable for security of the construction ($\ell < \log q$). So, it doesn't even make sense to use to try to use their result as a heuristic as the conditions are so far from being met.

Alternatively, we could observe that since $\|\mathbf{S}\|_x$ is small, in which case it is never reduced wrt. the modulus, we could wish to apply the results of [3], [1] which apply over infinite domains to argue that for $\mathbf{X} \leftarrow D^{n\times m}_{\mathbb{Z},\sigma}, \boldsymbol{y} \leftarrow D^m_{\mathbb{Z},\sigma'}$, it holds that $\mathbf{X}\boldsymbol{y} \approx_s D^n_{\mathbb{Z},\sigma'\mathbf{X}^T}$. Although, in that case we would at the very least require that $\mathbf{S}_{i,b}$ have dimensions $(n\log q)\times(n\log q)$. We could choose to increase the dimensions of the $\mathbf{S}_{i,b}$ to exploit this form, but leveraging the theorem in [1] would require each consecutive pair of $\mathbf{S}_i$ matrices to grow considerably in their standard deviation. This implies issues in managing error growth and obtaining small enough approximation factors for the underlying LWE assumption — possibly in satisfying other parameter relationships as well.

So, we have no promising direction to take to prove the conditional one-wayness required for the NIZK application in Construction 4.1.1. Next, we examine an alternative direction which we could take which accomplishes a similar goal.

7.2 Context-Hiding

The ephemeral randomness r_i which is added to the inputs in Construction 4.1.1, is added only in order to rely on conditional one-wayness. Since by definition PRFs are deterministic, in a worst-case situation in which a witness w for some statement x is unique — we can not hope for zero-knowledge. Appending r_i as a $O(\lambda)$ "salt" to inputs to makes the number of possible image points exponential in the security parameter.

Though, the core of the idea is that PRF outputs reveal nothing about their respective inputs, even given the secret key sk. So, we instead consider a more abstract notation

of *context-hiding* PRF outputs. In this case we would like that the prover could produce values which are testable, but not necessarily in the range of the PRF. In particular, these testable outputs should be provably independent of the input. We will formalize this by defining *simulated* testable outputs y' which can be produced without knowing an input such that $C(x) = b$, and whose distribution is close to values produced by PCT-PRF evaluation. For this idea to be useful, an evaluator should be able to produce a testable value in the same range as whatever they obtained from an honest evaluation — without compromising security. This is captured by the properties below.

$\mathsf{Hide}_{\mathsf{pp}}(y)$**:** is an efficient PPT algorithm that takes as input a PRF output $y \in \mathcal{R}_b$, and outputs a value $y' \in (\mathcal{Y} \times \mathcal{Y}) : \mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, y', b) = 1$.

For security, we require that there exists a simulator $\mathcal{S}.\mathsf{Hide}_{\mathsf{pp}}$ that for any choice of parameters produces a simulated $\tilde{y}'$ which is testable, but produced independently of any input x. We then require that the output of the simulator be indistinguishable from the output of $\mathsf{Hide}_{\mathsf{pp}}(\cdot)$. In particular, the following distributions should be computationally close for all PCT-PRFs $\mathsf{P} = (\mathsf{sk}, \mathsf{pp})$, all $C \in \mathcal{C}, \mathsf{sk}[C], x \in \mathcal{X}, b = C(x), y = \mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{sk}[C], x)$ and any efficient function f:

$$\left(\begin{array}{c} y, \\ y' = \mathsf{Hide}_{\mathsf{pp}}(y), \\ \mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y') \\ f(\mathsf{sk}, y') \end{array}\right) \approx_c \left(\begin{array}{c} y, \\ \tilde{y}' = \mathcal{S}.\mathsf{Hide}_{\mathsf{pp}}(b), \\ \mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, \tilde{y}') \\ f(\mathsf{sk}, \tilde{y}') \end{array}\right) \tag{7.2.1}$$

This property could be used in place of the salt and conditional one-wayness required in Lemma 4.1.4, and used to repeat an alternate proof for Construction 4.1.1. We will now show that the PCT-PRF in Construction 6.3.1 can support context-hiding.

7.2.1 Instantiating Context-Hiding

Comments Regarding Testability. We first observe some non-obvious properties which are permitted by the testability properties in Definition 3.1.1.

1. It is allowed that $\exists y \in \mathcal{Y} \setminus (\mathcal{R}_0 \cup \mathcal{R}_1), b \in \{0,1\} : \mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y) = 1$

2. For example, let $y' = [\mathbf{R}' \| \mathbf{R}'\mathbf{Z}_b]$ for $\mathbf{R}' \xleftarrow{\$} \mathbb{Z}_q^{n \times m/2}$, this trivially passes $\mathsf{Test}_{\mathsf{pp}}(\mathsf{sk}, b, y')$ as defined in Construction 6.3.1 for any $\mathbf{R}'$.

We may also clarify a potential weakness in the testing algorithm. For maliciously chosen small normed values, then tests for both values of $b \in \{0, 1\}$ pass and unique testability does not hold. Honest evaluations will not have such a small ℓ_∞ norm with high probability, so we could choose to reject these inputs. However, this is equivalent to outputting $\perp$ if *both* tests relative to 0 and 1 pass in the construction, since the norm bound on such malicious values means that their product will be small when multiplied with either preimage whp. This is the strategy used for Construction 4.1.1.

Recall that the matrices $\mathbf{A}_b^*$ are not output as part of the public parameters, but there is nothing that prevents us from doing so since we can invoke LWE to argue the outputs of the PRF remain indistinguishable from uniform — so even given $\mathbf{A}_b^*$, which range a value lies in remains hidden[8]. We can use this to exploit the randomized-self reducibility property of LWE shown in [56] Lemma 4.1. Regev observes that the bilinearity of inner products maps any sample from an LWE distribution under some public matrix $\mathbf{A}$ with secret $\boldsymbol{s}$, to a sample under $\mathbf{A}$ with secret $\boldsymbol{s} + \boldsymbol{t}$ for any $\boldsymbol{t} \in \mathbb{Z}_q^n$ (and maps to uniform otherwise).

So, we include $(\mathbf{A}_0^*, \mathbf{A}_1^*)$ as part of pp. Then, for any *unrounded* output $\hat{y} = \mathbf{S}_x\mathbf{A}_b^* + \mathbf{E}_x$ the evaluator samples $\mathbf{T} \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$. and outputs:

$$y' = \lfloor \hat{y} + \mathbf{T}\mathbf{A}_b^* \rceil_p = \lfloor (\mathbf{S}_x + \mathbf{T})\mathbf{A}_b^* + \mathbf{E}_x \rceil_p \tag{7.2.2}$$

Of course, by the security properties of the PCT-PRF, they are unable to tell *which* matrix to multiply with, but they can repeat the process once with each $\mathbf{A}_b^*$. For the correct choice of $\mathbf{A}_b^*$, the resulting sample has its secret term mapped to a uniformly random value $\mathbf{S}_x + \mathbf{T}$, which hides all information about $\mathbf{S}_x$. For the repetition under which the wrong $\mathbf{A}_b^*$ is used for the shift, the evaluator produces $\lfloor \mathbf{S}_x\mathbf{A}_b^* + \mathbf{E}_x + \mathbf{T}\mathbf{A}_{1-b}^* \rceil_p$. If q is prime, $\mathbf{T}\mathbf{A}_{1-b}^*$ is a uniformly random value, and sends the sample to the uniform distribution over $\mathbb{Z}_q^{n \times m}$, which of course also hides $\mathbf{S}_x$ as desired.

All of these steps can instead be done by adding $\lfloor \mathbf{T}\mathbf{A}_b^* \rceil_p$ to the rounded PRF output as well, since rounding is approximately linear up to some small binary error vector which has negligible impact on the testing procedure. We now define the context hiding algorithm and simulator, and show they satisfy the properties laid out in Section 7.2.

[8] This design rules out statements of the form "Given $\mathsf{sk}[C]$, it is difficult to find $y \in \mathcal{R}_b$ without knowing $x : C(x) = b$" as this is becomes trivial. So, any application which requires context hiding must rely on amplifying in parallel for a set of pseudorandom target ranges as done in Construction 4.1.1.

$\mathsf{P.Hide_{pp}}(y)$**:** Given a (rounded) output in $\mathcal{Y}$, sample $\mathbf{T} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, and output:

$$y' = (y + \lfloor \mathbf{T}\mathbf{A}_0^* \rceil_p, y + \lfloor \mathbf{T}\mathbf{A}_1^* \rceil_p)$$

$\mathcal{S}.\mathsf{Hide_{pp}}(b)$**:** Sample $\mathbf{T}, \mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{n \times m}$, and output:

$$\tilde{y}' = (\lfloor \mathbf{T}\mathbf{A}_b^* \rceil_p, \mathbf{U})$$

The algorithms above give $\tilde{y}' \approx_c y'$ by LWE with a small loss in advantage incurred by the ordering of the tuple. Similarly, it follows immediately from LWE that the evaluator learns nothing by running the $\mathsf{Hide_{pp}}$ algorithm. Since the context hiding does not affect the functionality of the trapdoor, for the entry in y' which gets sent to the correct distribution, testability is preserved. For the final condition, we have that $\mathbf{S}_x + \mathbf{T} \approx_s \mathbf{U}$ for any $\mathbf{S}_x$, and $\mathbf{T}, \mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{n \times m}$, which is stronger than the computational requirement on $f(\mathsf{sk}, \cdot)$.

7.3 Feasibility of the NIZK Application

The construction we achieve in Construction 6.3.1 only achieves the weaker simulation security of Definition 6.3.1, which does not allow testing during the security experiment. The proof strategy for Construction 4.1.1 relies on testability since it follows a similar proof strategy to those in [57], and as such we can not demonstrate a "bottom to top" provably secure NIZK. Similarly, as shown before we can not provably demonstrate conditional one-wayness is satisfied by the construction.

However, we can instantiate the rest of the components of the scheme since there are puncturable PRFs from lattice assumptions which can be evaluated in NC^1 and fit within the restrictions of the constraints supported by the scheme. Also, we have that for any $\mathcal{L} \in \mathsf{NP}$, $\mathcal{L} \leq \mathsf{3\text{-}SAT}$. This is relevant as $\mathsf{3\text{-}SAT}$'s witness relation $\mathcal{R}$ satisfies $\mathcal{R} \in \mathsf{NC}^1$. Finally, as shown above we can successfully construct a context-hiding extension to the scheme, which at least would suffice for the zero-knowledge component of the proof given the same proof strategy.

7.4 Testability of other PC-PRFs

Following [16], it was later demonstrated in [9] that the form of the PRF could be generalized further. By using public matrices of the form $\mathbf{A}_b = \mathbf{G}^{-1}(\mathbf{A}'_b)$ for $\mathbf{A}'_b$ uniform

over $\mathbb{Z}_q^{n \times m}$, the generalized scheme could could be proven using a more conventional LWE assumption. (Where $\mathbf{G}$ is a special structured matrix with a public trapdoor) The generalized construction also had revised requirements on the ratio p/q, and a more general algorithm for producing $\mathbf{A}_x$.

After further study, it was shown that the [9] style constructions could be combined with techniques from Attribute-based and Fully Homomorphic Encryption in [25] which also leveraged useful properties of the matrix $\mathbf{G}$ for supporting circuit evaluation. This led to circuit constrained PRFs, and finally Privately Constrained PRFs in [22] and [54]. For some constraint predicate C, these constructions produce outputs which essentially take the following form:

$$\mathsf{F}_{\boldsymbol{s},C}(x) = \left\lfloor \boldsymbol{s}^T \cdot (\mathbf{A}_x + C(x) \cdot \mathbf{G}) + \boldsymbol{e}^T \right\rceil_p \tag{7.4.1}$$

In these schemes, a PRF output lands near a fixed point in an unknown lattice induced by the input x, or a fixed coset of $\mathbf{A}_x$ induced by $\boldsymbol{s}^T\mathbf{G}$. The error term arises from the use of FHE schemes in the constructions which are used to hide the constraint circuit. Unfortunately, it is not clear how to test such a value even given $\boldsymbol{s}$ if the input is not known in advance (otherwise testing is trivial by subtracting $\lfloor \boldsymbol{s}^T(\mathbf{A}_x + C(x) \cdot \mathbf{G}) \rceil_p$ and checking the norm). When looking at the PC-PRF constructions in detail, it is also not clear how to modify them to introduce testability while preserving the properties they rely on for security and correctness.

7.5 Relation to other GGH15 Constructions

Recall that the Canetti and Chen Privately Constrained PRF ([27]) used preimage encodings of the following form to embed branching programs.

$$\mathbf{A}_i\mathbf{D}_{i,b} = (\mathbf{P}_{i,b} \otimes \mathbf{S}_{i,b}) \cdot (\mathbf{I}_{nw} \cdot \mathbf{A}_{i+1}) + \mathbf{E}_{i,b} \tag{7.5.1}$$

Independently, Goyal et al. [43] presented a related way of encoding branching programs using GGH15 for different applications (e.g. "Lockable Obfuscation"). This strategy was was later proven semantically secure in [42], and [58]. In particular, they sample a preimage for a block-permutations of each adjacent $\mathbf{A}$ matrix in the graph, and encode the

same secret into each block for each index i, b and rely on a standard normal-form LWE assumption, i.e.,

$$\mathbf{A}_i\mathbf{D}_{i,b} = (\mathbf{I}_w \otimes \mathbf{S}_{i,b}) \cdot (\mathbf{P}_{i,b} \otimes \mathbf{I}_n \cdot \mathbf{A}_{i+1}) + \mathbf{E}_{i,b}. \tag{7.5.2}$$

We can show that Eq. (7.5.1) and Eq. (7.5.2) are in fact equivalent due to the mixed-product property of the Kronecker product, by switching the positions of $\mathbf{P}_{i,b}$, and $\mathbf{I}_w$ since they have the same dimensions. This implies that LWE with this form of structured $\mathbf{S}$ (or structured $\mathbf{A}$) is equally as hard as normal-form LWE directly. The translation from one form to the other is as follows:

$$\begin{aligned}
(\mathbf{P} \otimes \mathbf{S}) \cdot \mathbf{A} + \mathbf{E} &= (\mathbf{P} \otimes \mathbf{S}) \cdot \mathbf{I}_{nw} \cdot \mathbf{A} + \mathbf{E} && (7.5.3)\\
&= (\mathbf{P} \otimes \mathbf{S}) \cdot (\mathbf{I}_w \otimes \mathbf{I}_n) \cdot \mathbf{A} + \mathbf{E} && (7.5.4)\\
&= (\mathbf{P} \cdot \mathbf{I}_w) \otimes (\mathbf{S} \cdot \mathbf{I}_n) \cdot \mathbf{A} + \mathbf{E} && (7.5.5)\\
&= (\mathbf{I}_w \otimes \mathbf{S}) \cdot (\mathbf{P} \otimes \mathbf{I}_n) \cdot \mathbf{A} + \mathbf{E}, && (7.5.6)
\end{aligned}$$

where the transformation follows from the mixed-product property of the Kronecker product and commutativity of identity matrices.

Canetti and Chen used the GGH15 branching program techniques to create a privately constrained PRF for NC^1, however their construction as designed, does not admit testability. Notice that Definition 2.2.3 requires the output of the PRF to be indistinguishable from uniform for any $x : C(x) = 1$. For any GGH15 encoding defined as above, we cannot satisfy this condition when constructing Privately Constrained PRFs. Recall that all branching programs of a fixed length share the same input to index map, and the same output permutation $\mathbf{P}^*$ — therefore we consider $\mathbf{P}^*$ to be public. Since a GGH15 evaluation produces the entire output of the final permutation, we can easily map any output for $C(x) = 0$, to the output $C(x) = 1$ by appropriately inverting the permutation $\mathbf{P}^*$. This yields a trivial attack on residual pseudorandomness.

Canetti and Chen circumvented this problem by introducing a lossy map (a "bookend" in iO literature) such that the attack sketched above is impossible. In particular, they "fold" a matrix $\mathbf{J} \in \chi^{n \times nw}$ into $\mathbf{A}_1$ by encoding it in an additional high dimensional LWE term. That is, instead of publishing $\mathbf{A}_1$, they release $\mathbf{J}\mathbf{A}_1 + \mathbf{E}'$. Then, the analysis from Theorem 6.2.2 carries over. Since the resulting output is lossy, the permutation cannot be inverted without first recovering $\mathbf{A}_1$, which implies breaking LWE. Then, they demonstrated an expression in

terms of $\mathbf{J}$ produced during the evaluation is pseudorandom by reduction to LWE, which lets them use this as the key in the form of the PRF in Eq. (6.1.3) to argue pseudorandomness.

An easier strategy which accomplishes the same result is to simply drop all but the first n of the rows of $\mathbf{A}_0$ (as in Construction 6.3.1, which is used implicitly in [42] and [58] for correctness. Of course, this map is also lossy, so we can hope to use it in an updated construction that does admit testability. The analysis of pseudorandomness is then derived from a closer inspection of the structure induced by the branching evaluation, as in Theorem 6.2.3. [27] was later updated to also include this simplified lossy map in their construction, and becomes equivalent to Construction 6.3.1 aside from testability and the two-sided residual pseudorandomness. This strategy is also present in the Privately Constrained PRF construction of [31] (which is otherwise notably different), although their construction can only be made testable for the *constrained range*.

7.6 One-sided Testability

The testability of Definition 3.1.1 requires that outputs in both ranges be testable, and is defined in terms of a "two-sided" residual pseudorandomness property. We could instead define these properties such that *only* the constrained range (e.g. the image under the constrained key $\mathsf{sk}[C]$ for all authorized inputs $x : C(x) = 1$) is testable. That is, the constrained range is distinguishable from random given the secret key. This variant is relevant to consider with respect to actual constructions. Recall that correctness guarantees that the range of the PRF under the $\mathsf{sk}[C]$ coincides with the range under sk for authorized inputs. So, if we assume the outputs of sk are testable given sk, then outputs produced using $\mathsf{sk}[C]$ on authorized inputs are also testable.

The definition of the standard privately constrained PRF in Definition 2.2.3 only requires that the range of the function under $\mathsf{sk}[C]$ for all *unauthorized* inputs is indistinguishable from random — so the images of these points may deviate from the image under sk in some unpredictable manner. Then, it is not necessarily true that there exists *any* efficient algorithm which can *also* test outputs of the constrained key for *unauthorized* inputs. We can recover completeness in this setting for predicate constraints by giving out constrained keys for a circuit and its complement: $\mathsf{sk}[C], \mathsf{sk}'[\bar{C}]$ (with respect to different secret keys to avoid the iO implications of Remark 2.2.1).

A construction of a testable PRF with the above limitation admits an efficient attack on any scheme using the primitive. If the PRF has a large expansion factor, *and* satisfies computational testing completeness — we have a trivial attack on residual pseudorandomness.

Namely, given the output y of a PRF which is to be tested, an adversary can perform an arbitrary perturbation of the value (flip a bit, or even replace it with a uniformly random value) to obtain y', and send y' to a challenger with testing capability.

If the behaviour of the challenger is unaffected, this implies y was not in the testable range. Otherwise, if the behaviour of the challenger does change, this implies y had to have been in the testable range. In either case, residual pseudorandomness is not satisfied.

Chapter 8

Conclusions and Future Work

This work defined and initiated the study of a new cryptographic primitive: Privately Constrained *Testable* PRFs. Unfortunately, our understanding of this primitive is still quite limited. We are able to construct the primitive from lattice assumptions, but only prove it is secure in a security model which removes a core component of the functionality. So, we are currently unable to achieve a "top to bottom" provably secure instantiation of any application.

Additionally, the proof strategies which seem to follow most naturally (e.g. the soundness proof of the NIZK scheme in Theorem 4.1.2) from our stronger definition appear to require that the primitive behave as an interactive kind of "Indistinguishability Obfuscation" (iO). This is intuitively a much stronger assumption, since our primitive only allows outputs to be learnable for the holder of the secret key. This strongly suggests that the definitions and proof strategies presented are not the correct approach to yielding usable constructions.

Alternatively, the gap encountered in the NIZK soundness proof is simply an artifact of the proof technique, and there are tools that can be leveraged to make a different proof go through. Depending on the particular application, it may be possible to prove security by relying on parallelization, weaker combinatorial techniques and/or complexity leveraging. One possibility is to dramatically reduce testing soundness such that uniformly random values always output *some* valid bit during testing, and rely on threshold techniques to mitigate distinguishing advantage gained from intermediate losses of correctness. However, the viability of this idea is inconclusive without formal analysis.

We conclude with the following open problems:

1. Formulate an alternate definition for Privately Constrained Testable PRFs which

admits provably secure applications and an instantiation of the primitive in that model.

2. The GGH15 strategy poses a tremendous barrier to any indistinguishability-based definition, such that resolving the problem would lead to a significant step towards achieving iO. Find an instantiation of privately constrained testable PRFs which does not require using GGH15 such that testability is preserved concurrently with circuit indistinguishability. Furthermore, find a construction for which the design of the testability and residual pseudorandomness are not mutually exclusive (as in our context-hiding construction).

3. Analyze additional applications of the primitive, and determine whether they can be proven secure in the security models in which we have been able to instantiate the primitive.

References

[1] Divesh Aggarwal and Oded Regev. A note on discrete gaussian combinations of lattice vectors. *CoRR*, abs/1308.2405, 2013.

[2] Shweta Agrawal. New Methods for Indistinguishability Obfuscation: Bootstrapping and Instantiation. Cryptology ePrint Archive, Report 2018/633, 2018. https://eprint.iacr.org/2018/633.

[3] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete Gaussian Leftover Hash Lemma over Infinite Domains. Cryptology ePrint Archive, Report 2012/714, 2012. https://eprint.iacr.org/2012/714.

[4] D. Aharonov and O. Regev. Lattice problems in $NP \cap coNP$. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 362–371, Oct 2004.

[5] M. Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[6] Joel Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with Rounding, Revisited: New Reduction, Properties and Applications. Cryptology ePrint Archive, Report 2013/098, 2013. https://eprint.iacr.org/2013/098.

[7] Prabhanjan Ananth, Aayush Jain, Dakshita Khurana, and Amit Sahai. Indistinguishability Obfuscation Without Multilinear Maps: iO from LWE, Bilinear Maps, and Weak Pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. https://eprint.iacr.org/2018/615.

[8] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '09, pages 595–618, Berlin, Heidelberg, 2009. Springer-Verlag.

[9] Abhishek Banerjee and Chris Peikert. New and Improved Key-Homomorphic Pseudorandom Functions. Cryptology ePrint Archive, Report 2014/074, 2014. https://eprint.iacr.org/2014/074.

[10] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom Functions and Lattices. Cryptology ePrint Archive, Report 2011/401, 2011. https://eprint.iacr.org/2011/401.

[11] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 1–18, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[12] D A Barrington. Bounded-width Polynomial-size Branching Programs Recognize Exactly Those Languages in NC1. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 1–5, New York, NY, USA, 1986. ACM.

[13] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafael del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-Linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits. Cryptology ePrint Archive, Report 2018/560, 2018. https://eprint.iacr.org/2018/560.

[14] Dan Boneh, Sam Kim, and Hart Montgomery. Private Puncturable PRFs From Standard Lattice Assumptions. Cryptology ePrint Archive, Report 2017/100, 2017. https://eprint.iacr.org/2017/100.

[15] Dan Boneh, Sam Kim, and David J. Wu. Constrained Keys for Invertible Pseudorandom Functions. Cryptology ePrint Archive, Report 2017/477, 2017. http://eprint.iacr.org/2017/477.

[16] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key Homomorphic PRFs and Their Applications. Cryptology ePrint Archive, Report 2015/220, 2015. https://eprint.iacr.org/2015/220.

[17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining Pseudorandom Functions Privately. Cryptology ePrint Archive, Report 2015/1167, 2015. https://eprint.iacr.org/2015/1167.

[18] Dan Boneh and Brent Waters. Constrained Pseudorandom Functions and Their Applications. Cryptology ePrint Archive, Report 2013/352, 2013. https://eprint.iacr.org/2013/352.

[19] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional Signatures and Pseudorandom Functions. Cryptology ePrint Archive, Report 2013/401, 2013. https://eprint.iacr.org/2013/401.

[20] Zvika Brakerski, David Cash, Rotem Tsabary, and Hoeteck Wee. Targeted Homomorphic Attribute Based Encryption. Cryptology ePrint Archive, Report 2016/691, 2016. https://eprint.iacr.org/2016/691.

[21] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical Hardness of Learning with Errors. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, New York, NY, USA, 2013. ACM.

[22] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private Constrained PRFs (and More) from LWE. Cryptology ePrint Archive, Report 2017/795, 2017. http://eprint.iacr.org/2017/795.

[23] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. Cryptology ePrint Archive, Report 2011/344, 2011. https://eprint.iacr.org/2011/344.

[24] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-Based FHE as Secure as PKE. Cryptology ePrint Archive, Report 2013/541, 2013. https://eprint.iacr.org/2013/541.

[25] Zvika Brakerski and Vinod Vaikuntanathan. Constrained Key-Homomorphic PRFs from Standard Lattice Assumptions Or: How to Secretly Embed a Circuit in Your PRF. Cryptology ePrint Archive, Report 2015/032, 2015. https://eprint.iacr.org/2015/032.

[26] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating Conjunctions Under Entropic Ring LWE. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 147–156, New York, NY, USA, 2016. ACM.

[27] Ran Canetti and Yilei Chen. Constraint-hiding Constrained PRFs for NC1 from LWE. Cryptology ePrint Archive, Report 2017/143, 2017. http://eprint.iacr.org/2017/143.

[28] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 523–552, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[29] Yilei Chen. An Alternative View of the Graph-Induced Multilinear Maps. Cryptology ePrint Archive, Report 2016/200, 2016. https://eprint.iacr.org/2016/200.

[30] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of Candidate Branching Program Obfuscators. Cryptology ePrint Archive, Report 2016/998, 2016. https://eprint.iacr.org/2016/998.

[31] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 Beyond Permutation Branching Programs: Proofs, Attacks, and Candidates. Cryptology ePrint Archive, Report 2018/360, 2018. https://eprint.iacr.org/2018/360.

[32] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 Multilinear Maps. Cryptology ePrint Archive, Report 2015/1037, 2015. https://eprint.iacr.org/2015/1037.

[33] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[34] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-Induced Multilinear Maps from Lattices. Cryptology ePrint Archive, Report 2014/645, 2015. https://eprint.iacr.org/2014/645.

[35] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. Cryptology ePrint Archive, Report 2007/432, 2008. https://eprint.iacr.org/2007/432.

[36] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. Cryptology ePrint Archive, Report 2013/340, 2013. https://eprint.iacr.org/2013/340.

[37] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, August 1986.

[38] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable Garbled Circuits and Succinct Functional Encryption. Cryptology ePrint Archive, Report 2012/733, 2012. https://eprint.iacr.org/2012/733.

[39] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled Fully Homomorphic Signatures from Standard Lattices. Cryptology ePrint Archive, Report 2014/897, 2014. https://eprint.iacr.org/2014/897.

[40] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on Asymmetry: Efficient ABE for Branching Programs. Cryptology ePrint Archive, Report 2014/819, 2014. https://eprint.iacr.org/2014/819.

[41] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A Generic Approach to Constructing and Proving Verifiable Random Functions. Cryptology ePrint Archive, Report 2017/021, 2017. https://eprint.iacr.org/2017/021.

[42] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable Obfuscation. Cryptology ePrint Archive, Report 2017/274, 2017. https://eprint.iacr.org/2017/274.

[43] Rishab Goyal, Venkata Koppula, and Brent Waters. Separating Semantic and Circular Security for Symmetric-Key Bit Encryption from the Learning with Errors Assumption. Cryptology ePrint Archive, Report 2017/120, 2017. https://eprint.iacr.org/2017/120.

[44] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 248–253, Washington, DC, USA, 1989. IEEE Computer Society.

[45] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, Sep 1996.

[46] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable Pseudorandom Functions and Applications. Cryptology ePrint Archive, Report 2013/379, 2013. https://eprint.iacr.org/2013/379.

[47] Sam Kim and David J. Wu. Multi-Theorem Preprocessing NIZKs from Lattices. Cryptology ePrint Archive, Report 2018/272, 2018. https://eprint.iacr.org/2018/272.

[48] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[49] Daniele Micciancio. Almost Perfect Lattices, the Covering Radius Problem, and Applications to Ajtai's Connection Factor. *SIAM J. Comput.*, 34(1):118–169, January 2005.

[50] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. Cryptology ePrint Archive, Report 2011/501, 2012. https://eprint.iacr.org/2011/501.

[51] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[52] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Relations Among Notions of Non-malleability for Encryption. In *Proceedings of the Advances in Crypotology 13th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT'07, pages 519–535, Berlin, Heidelberg, 2007. Springer-Verlag.

[53] Chris Peikert. A Decade of Lattice Cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, March 2016.

[54] Chris Peikert and Sina Shiehian. Privately Constraining and Programming PRFs, the LWE Way. Cryptology ePrint Archive, Report 2017/1094, 2017. https://eprint.iacr.org/2017/1094.

[55] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic Function Evaluation and Applications. Cryptology ePrint Archive, Report 2018/409, 2018. https://eprint.iacr.org/2018/409.

[56] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[57] Amit Sahai and Brent Waters. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More. Cryptology ePrint Archive, Report 2013/454, 2013. https://eprint.iacr.org/2013/454.

[58] Daniel Wichs and Giorgos Zirdelis. Obfuscating Compute-and-Compare Programs under LWE. Cryptology ePrint Archive, Report 2017/276, 2017. https://eprint.iacr.org/2017/276.