TÉCNICO LISBOA

Private Constrained Pseudorandom Functions with Succinct Keys

Pedro Rocha Peixoto Capitão

Thesis to obtain the Master of Science Degree in

Mathematics and Applications

Supervisors: Paulo Alexandre Carreira Mateus
Pedro de Melo Branco

Examination Committee

Chairperson: Maria Cristina de Sales Viana Serôdio Sernadas
Supervisor: Paulo Alexandre Carreira Mateus
Member of Committee: Lisa Maria Kohl

July 2021

Acknowledgments

I would like to thank my supervisors, Pedro Branco and Paulo Mateus, who guided and accompanied me in all the stages of this project, from its inception to its conclusion and through all the difficulties in between. It was a pleasure to work with them.

I also wish to thank Lisa Kohl for taking the time to read an earlier version of this thesis and providing helpful comments and suggestions.

Lastly, I thank Mariana and my family for their invaluable support. I am very lucky to have them.

Resumo

Uma função pseudo-aleatória constrangida é uma função pseudo-aleatória (PRF) que permite derivar chaves constrangidas a partir da chave mestra. Cada chave constrangida está associada a um constrangimento f e permite avaliar a PRF em pontos x satisfazendo $f(x) = 0$, mas não dá qualquer informação sobre os valores da PRF nos pontos x tais que $f(x) = 1$. Numa PRF constrangida privada, as chaves constrangidas não revelam os constrangimentos que lhes correspondem.

Nesta tese consideramos o problema de construir uma PRF constrangida privada tal que o tamanho das chaves constrangidas seja independente dos constrangimentos que lhes correspondem. Mostramos que isto é possível quando usamos uma definição generalizada de PRF constrangida, segundo a qual os parâmetros públicos podem ser atualizados sempre que é gerada uma chave. Propomos duas construções distintas que cumprem este objetivo, partindo de uma PRF constrangida privada e usando como ferramentas as primitivas criptográficas *attribute-based encryption* e *functional encodings*, respetivamente. A sua segurança baseia-se na dificuldade do problema *learning with errors*, que está relacionado com problemas em reticulados muito estudados e para os quais não se conhecem algoritmos eficientes.

Palavras-chave: criptografia em reticulados, learning with errors, função pseudo-aleatória constrangida, constrangimentos privados, parâmetros atualizáveis.

Abstract

A constrained pseudorandom function is a pseudorandom function (PRF) in which constrained keys can be derived from the master secret key. Each constrained key is associated with a constraint f and allows its user to evaluate the PRF at points x satisfying $f(x) = 0$, but gives no information about the PRF values at points x such that $f(x) = 1$. In a private constrained PRF, constrained keys do not reveal their corresponding constraints.

In this thesis we consider the problem of building a private constrained PRF in which the size of the constrained keys is independent of the constraints. We show that this is possible to achieve under a generalized definition of constrained PRF, in which the public parameters may be updated whenever a key is generated. We provide two distinct constructions that fulfill these requirements, starting from a private constrained PRF and using as tools the cryptographic primitives of attribute-based encryption and functional encodings, respectively. Their security is based on the hardness of the learning with errors problem, which is related to the intractability of well-studied problems on lattices.

Keywords: lattice cryptography, learning with errors, constrained pseudorandom function, private constraints, updatable parameters.

Contents

Acknowledgments iii

Resumo v

Abstract vii

List of Abbreviations xi

1 Introduction 1

2 Constrained pseudorandom functions 5

2.1 Pseudorandom functions 5

2.2 Constrained pseudorandom functions 6

2.3 Updatable parameters 8

2.4 Applications 11

2.4.1 Database with restricted access 11

2.4.2 Watermarking 11

3 The BTVW private constrained PRF 13

3.1 Lattices and learning with errors 13

3.2 Homomorphic evaluation over matrices 14

3.3 Fully homomorphic encryption 15

3.4 LWE-based private constrained PRF 16

4 Succinct-key private constrained PRF from attribute-based encryption 19

4.1 Attribute-based encryption 19

4.2 Succinct-key private constrained PRF 20

4.3 Proof of security 22

4.4 Size of constrained keys 27

4.5 Separation between weak and strong security 28

5 Succinct-key private constrained PRF from functional encodings 29

5.1 Symmetric-key encryption 29

5.2 Functional encodings 30
5.3 Succinct-key private constrained PRF 31
5.4 Proof of security 32
5.5 Size of constrained keys 36

6 Conclusions **39**

Bibliography **43**

List of Abbreviations

ABE	Attribute-based encryption
CPRF	Constrained pseudorandom function
FE	Functional encodings
FHE	Fully homomorphic encryption
LWE	Learning with errors
PPT	Probabilistic polynomial time
PRF	Pseudorandom function
SKE	Symmetric-key encryption

Chapter 1

Introduction

Many cryptographic protocols used in practice today are vulnerable to attacks by quantum algorithms. Most notably, the security of many cryptosystems relies on the computational hardness of number-theoretic problems such as integer factorization or the discrete logarithm, which are efficiently solved by Shor's algorithm [Sho97]. Widely used public-key cryptosystems, such as RSA, are among those susceptible to quantum attacks.

Quantum attacks on cryptosystems cannot currently be realized because quantum computers with sufficient processing power do not exist. However, there has recently been an increasing investment in the development of this technology and it is generally believed that large-scale quantum computers will become a reality in the near future. The threat of the emergence of quantum computers has sparked a large interest in post-quantum cryptography, the field of research concerning cryptosystems with security against quantum attacks. For instance, the National Institute of Standards and Technology (NIST) is currently conducting their program Post-Quantum Cryptography Standardization, a process to standardize quantum-resistant public-key cryptosystems, in which the majority of submissions were lattice-based systems.

Lattice cryptography is the study of cryptographic protocols provably secure under the assumption of intractability of certain computational problems on lattices. Such problems are conjectured to not be efficiently solvable by classical or quantum algorithms, making these systems quantum-resistant. But security against quantum attacks is not the only noteworthy quality of lattice cryptography. The problem of building fully-homomorphic encryption, a powerful cryptographic primitive which allows computation over encrypted data, remained largely unanswered for thirty years until the work of Gentry [Gen09], which used lattices to give the first candidate construction based on cryptographic assumptions. Lattice cryptography has provided the first (and, in some cases, all) constructions of this and other useful cryptographic primitives such as attribute-based encryption for arbitrary constraints [GVW13] or indistiguishability obfuscation [GGH$^+$13]. Moreover, lattice-based protocols are generally simple, efficient and parallelizable. The cryptographic protocols studied in this thesis rely on the hardness of the learning with errors (LWE) problem for their security. LWE was introduced by Regev [Reg05] and is a computational problem related to lattices with wide application in cryptography.

The subject of this thesis is the cryptographic protocol known as private constrained pseudorandom function. A pseudorandom function (PRF) [GGM86] is a keyed function such that, for a random key, its outputs are indistinguishable from those of a truly random function. In a constrained pseudorandom function (CPRF) [BW13, KPTZ13, BGI14], the owner can delegate constrained keys which allow other parties to evaluate the PRF. However, each key is associated with a constraint (a predicate over the domain of the PRF) and only evaluates the PRF correctly at points which satisfy that constraint.

A CPRF is said to be private (or constraint-hiding) [BLW17] if the constrained keys reveal no information about the corresponding constraints. Constructions of private CPRFs for different classes of constraints have been proposed, including for point-functions [BKM17], for constraints in NC^1 [CC17], and for all polynomial-size circuits [BLW17, BTVW17, PS18]. All of these are based on LWE, with the exception of [BLW17], which requires the powerful cryptographic primitive of indistinguishability obfuscation [BGI$^+$12] and is also the only one that achieves collusion resistance.

A desirable property for CPRFs is that the constrained keys are short. Ideally, the size of such a key should be (asymptotically) independent from the constraint that is associated to it – when this is the case we say that the keys are succinct. An example of this is the LWE-based CPRF proposed by Brakerski and Vaikuntanathan [BV15], which has succinct constrained keys. However, the problem appears to be harder when we move from standard CPRFs to private CPRFs.

Our main contribution is a private CPRF with succinct keys for the constraint class of all polynomial-size circuits (with bounded depth). We provide two different constructions matching this description, both of which build upon a private CPRF [BTVW17] which supports the same constraint class, but in which the keys are not succinct. The first relies on the internal structure of this private CPRF and on the use of attribute-based encryption with short keys [BGG$^+$14]. The second uses functional encodings [WW21] and a private CPRF as a black box. We do not know of any previous private CPRF construction with succinct constrained keys.

There is a caveat to our results – in order to achieve the property of succinct keys, we resorted to a generalized definition of constrained PRF. In this new notion, which we call CPRF with updatable parameters, the public parameters of the scheme can be updated whenever a constrained key is generated (this is equivalent to generating both a public and a private constrained key). We argue that little is lost by considering this definition, as it is a simple and intuitive generalization and it can still be used in all applications of private CPRFs that we are aware of.

In terms of security, both of our constructions satisfy essentially the same definition as the underlying private CPRF, which is single-key selective security. This means that the scheme is resistant against an adversary that has access to only one constrained key, chosen at the start of the security game. While this is a relatively weak notion, there are indicators that achieving stronger security is substantially harder. For instance, it has been shown that a 2-key secure private CPRF implies the existence of indistinguishability obfuscation [CC17] and that a certain simulation-based definition of full (or adaptive) security is impossible to obtain [BKM17]. We define two generalizations of single-key selective security for CPRFs with updatable parameters. Our first construction satisfies only the weaker of the two definitions, while the second satisfies both.

The thesis is structured as follows.

In Chapter 2 we formally define CPRFs and private CPRFs. We also define CPRFs with updatable parameters and present some applications of private CPRFs with succinct keys.

Chapter 3 is dedicated to the private CPRF scheme of Brakerski, Tsabary, Vaikuntanathan, and Wee [BTVW17], which is the starting point for our proposed protocols. We start by giving some background on lattice-based cryptography and then, after presenting the necessary tools, we describe their construction.

Our two proposed private CPRF schemes are presented in Chapters 4 and 5. We describe each of them in detail and prove that they are correct, secure and have succinct constrained keys. Our main results are Theorems 4.4.2 and 5.5.3.

Finally, in Chapter 6 we briefly discuss the obtained results and suggest some possible directions for future work.

Chapter 2

Constrained pseudorandom functions

We begin by establishing some notation that will be used throughout the thesis. If χ is a probability distribution over a set X, the expression $x \leftarrow \chi$ denotes that x is a random variable with distribution χ, and, when X is finite, $x \leftarrow X$ denotes that the distribution of x is the uniform distribution on X. If Alg is a probabilistic algorithm, $y \leftarrow \mathsf{Alg}(x)$ indicates that y is a random variable distributed according to the output of Alg on input x. For a variable x considered as the input or output of some algorithm, $|x|$ denotes its size in bits. The expression $\mathsf{poly}(\cdot)$ denotes any polynomial function; for instance, $f(n) \leqslant \mathsf{poly}(n)$ means that f is bounded by some polynomial. Likewise, $\mathsf{negl}(\cdot)$ denotes a negligible function: a function f such that, for any positive polynomial p, $f(n) \leqslant \frac{1}{p(n)}$ for sufficiently large n. We use the standard asymptotic notation $\mathcal{O}(\cdot)$ and its variant $\tilde{\mathcal{O}}(\cdot)$ which ignores logarithmic factors in the indicated variable.

The expression $\log$ always denotes the logarithm in base 2. For a real number x, we denote by $\lceil x \rceil$ the smallest integer that is greater than or equal to x, and by $\lfloor x \rceil = \lceil x - \frac{1}{2} \rceil$ the integer closest to x. For a natural number q, the symbol $\mathbb{Z}_q$ represents the ring of integers modulo q. We also define $[n] = \{1, 2, \ldots, n\}$. If $\mathbf{A}_1, \mathbf{A}_2$ are matrices, $[\mathbf{A}_1 \,|\, \mathbf{A}_2]$ denotes their horizontal concatenation and $\left(\begin{smallmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{smallmatrix}\right)$ their vertical concatenation. We denote by $\|\mathbf{v}\|_2$ the Euclidean norm of a vector $\mathbf{v}$ and by $\|\mathbf{A}\|_\infty$ the largest absolute value of the entries of a matrix $\mathbf{A}$.

2.1 Pseudorandom functions

A pseudorandom function (PRF) with key space $\mathcal{K}$, domain $\mathcal{X}$ and range $\mathcal{Y}$ (where the size of these sets may depend on the security parameter λ) is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. The security requirement, called pseudorandomness, is that, for a uniform $k \leftarrow \mathcal{K}$, the outputs of the keyed function $F(k, \cdot)$ are computationally indistinguishable from the outputs of a uniformly chosen function from $\mathcal{X}$ to $\mathcal{Y}$.

Pseudorandom functions were introduced by Goldreich, Goldwasser and Micali [GGM86], who showed that they can be built from one-way functions. They have a wide array of applications and commonly serve as building blocks for more complex cryptographic protocols.

2.2 Constrained pseudorandom functions

In a constrained pseudorandom function (CPRF), the owner of the master secret key can generate constrained keys corresponding to constraints $f : \{0,1\}^z \to \{0,1\}$. Such a key allows its holder to compute the value of the PRF on points $x \in \{0,1\}^z$ such that $f(x) = 0$, i.e. x satisfies f.[1] Constrained PRFs were proposed independently by Boneh and Waters [BW13], Kiayias, Papadopoulos, Triandopoulos, and Zacharias [KPTZ13], and Boyle, Goldwasser, and Ivan [BGI14].

Following [BTVW17] and without loss of generality, we consider CPRFs with domain $\{0,1\}^z$, range $\mathbb{Z}_p$ and constraints represented by strings in $\{0,1\}^\ell$. All the CPRF constructions featured in this work are for the class of all constraints computable by Boolean circuits of polynomial size and a priori bounded depth t.

Definition 2.2.1 (Constrained PRF)**.** *A* constrained pseudorandom function *is a tuple* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ *of polynomial-time algorithms with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$ *is a probabilistic algorithm that receives as input the security parameter* λ*, the maximum description length* ℓ *of constraint functions, their input length* z *and their maximum depth* t*. It outputs a master secret key* msk *and public parameters* pp*.*

- $\mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$ *is a deterministic algorithm that receives as input a key* msk *and a string* $x \in \{0,1\}^z$*, and outputs a value* $y \in \mathbb{Z}_p$*.*

- $\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$ *is a probabilistic algorithm that receives as input a key* msk *and a circuit* $f : \{0,1\}^z \to \{0,1\}$*. It outputs a constrained key* ck*.*

- $\mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{ck}, x)$ *is a deterministic algorithm that receives as input a string* $x \in \{0,1\}^z$ *and a constrained key* ck*. It outputs a value* $y \in \mathbb{Z}_p$*.*

Correctness. For any $x \in \{0,1\}^z$ and $f \in \{0,1\}^\ell$ such that $f(x) = 0$,

$$\Pr\left[\mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x) = \mathsf{ConstrainEval}_{\mathsf{pp}}(\mathsf{ck}, x)\right] \geqslant 1 - \mathrm{negl}(\lambda),$$

where $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\mathsf{ck} \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$ and the probability is taken over the randomness of KeyGen and $\mathsf{Constrain}$.

A constrained PRF is secure if it is pseudorandom at constrained points. This means that a key corresponding to a constraint f gives no information about the PRF output at input points that do not satisfy f.

A stronger form of this protocol, named private constrained pseudorandom function, was introduced by Boneh, Lewi and Wu [BLW17]. These have the additional security requirement of constraint hiding (or privacy), which states that constrained keys should not reveal information about their corresponding constraints.

[1] Since we are working with the CPRF of [BTVW17], we follow their convention of writing $f(x) = 0$ when x satisfies f. Note that this is contrary to the more common convention in which f is said to be satisfied if $f(x) = 1$, but the two are equivalent.

We consider the indistinguishability-based notion of single-key selective security for private CPRFs. It comprises two properties, which we present separately for clarity, adapting the definition from [BTVW17]. A standard (i.e., non-private) CPRF is said to have single-key selective security if it satisfies only the first property.

Definition 2.2.2 (Security)**.** *A constrained PRF* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ *is a* single-key selective private constrained pseudorandom function *if the following conditions hold:*

- **Pseudorandomness at constrained points.** *Consider the following game between a challenger and a stateful probabilistic polynomial-time (PPT) adversary $\mathcal{A}$:*

  1. *$\mathcal{A}$ sends 1^ℓ, 1^t and $f \in \{0,1\}^\ell$ to the challenger.*
  2. *The challenger generates $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$ and $\mathsf{ck} \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$. It flips a coin $b \leftarrow \{0,1\}$ and sends $(\mathsf{pp}, \mathsf{ck})$ to $\mathcal{A}$.*
  3. *$\mathcal{A}$ can send queries $x \in \{0,1\}^z$ such that $f(x) = 1$, with no value x queried more than once. The challenger returns $y = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$, if $b = 0$, or $y \leftarrow \mathbb{Z}_p$, if $b = 1$.*
  4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

  Then $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ for any adversary $\mathcal{A}$.

- **Constraint hiding.** *Consider the following game between a challenger and a stateful PPT adversary $\mathcal{A}$:*

  1. *$\mathcal{A}$ sends 1^ℓ, 1^t and $f^0, f^1 \in \{0,1\}^\ell$ to the challenger.*
  2. *The challenger generates $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$ and a coin $b \leftarrow \{0,1\}$. It generates $\mathsf{ck} \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f^b)$ and sends $(\mathsf{pp}, \mathsf{ck})$ to $\mathcal{A}$.*
  3. *$\mathcal{A}$ can send queries $x \in \{0,1\}^z$ such that $f^0(x) = f^1(x)$, to which the challenger returns $y = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$.*
  4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

  Then $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ for any adversary $\mathcal{A}$.

The first property in the security definition states that the output of the PRF on a point x is indistinguishable from a random output, even for a holder of a constrained key for a predicate f such that $f(x) = 1$. The property of constraint hiding asserts that a constrained key ck does not reveal its corresponding constraint f. We say that the quantity $\left|\Pr[b' = b] - \frac{1}{2}\right|$ in the above games is the *advantage* of the adversary.

Succinct keys. We say that a constrained PRF has *succinct keys* if the size of each constrained key is asymptotically independent of the size of the description of the constraint, depending only on the security parameter λ. More specifically, the scheme has succinct keys if there exists a polynomial $p(\cdot)$ such that, for any polynomial $\ell(\cdot)$, if ck is a constrained key corresponding to $f \in \{0,1\}^{\ell(\lambda)}$, then $|\mathrm{ck}| \leqslant p(\lambda)$ for sufficiently large λ.

2.3 Updatable parameters

We define a generalized notion of constrained PRFs, in which the public parameters are updated whenever a constrained key is generated.

Syntax. A *constrained pseudorandom function with updatable parameters* is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ satisfying the conditions of Definition 2.2.1, with the following difference: the $\mathsf{Constrain}$ algorithm outputs a pair $(\mathsf{pp}', \mathsf{ck})$ consisting of new public parameters pp' and a constrained key ck.

Correctness. Let $f \in \{0,1\}^\ell$, $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and $(\mathsf{pp}', \mathsf{ck}) \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$. Then $\mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x) = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$ for any $x \in \{0,1\}^z$. Moreover, if $f(x) = 0$ then

$$\Pr\left[\mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x) = \mathsf{ConstrainEval}_{\mathsf{pp}'}(\mathsf{ck}, x)\right] \geqslant 1 - \mathrm{negl}(\lambda),$$

where the probability is taken over the randomness of KeyGen and $\mathsf{Constrain}$.

In the usual definition of a constrained PRF, the $\mathsf{Constrain}$ algorithm outputs only a constrained key. This definition includes that notion as the particular case in which the public parameters are not updated, i.e. $\mathsf{pp}' = \mathsf{pp}$. Note that in our constructions the updates are always incremental – if the parameters pp are replaced by pp', then pp' contains all the information present in pp. Therefore this new notion could equivalently be defined as a CPRF in which the constrained key is split into two parts: a private part (which corresponds to the actual constrained key) and a public part (the additional information added to the parameters).

When clear from the context, we will often omit the expression "with updatable parameters". Note that both of our proposed constructions, labelled SCPRF in Chapters 4 and 5, are private CPRFs with updatable parameters.

Regarding the size of constrained keys, the main advantage of this new definition is that it allows the $\mathsf{Constrain}$ algorithm to generate additional information that does not need to be succinct (because it becomes part of the parameters) while still having succinct keys. However, we need a new security definition that reflects the fact that the updated parameters are assumed to be public – otherwise any CPRF could trivially be made into a succinct-key CPRF with updatable parameters by placing the old constrained key in the updated parameters and letting the new constrained key be empty.

We define two different notions of security for private constrained PRFs with updatable parameters, both of which generalize Definition 2.2.2. In the first, which we call weak security, the adversary has access to updated public parameters that are generated for uniformly chosen constraints, unknown to the adversary. This corresponds to a real-world adversary waiting for the parameters to be updated as keys are generated for other users. It is a natural extension of Definition 2.2.2, in which there is no collusion resistance.

Definition 2.3.1 (Weak security)**.** *A constrained pseudorandom function with updatable parameters* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ *is a* weak single-key selective private constrained pseudorandom function *if the following conditions hold:*

- **Pseudorandomness at constrained points.** *Consider the following game between a challenger and a stateful PPT adversary $\mathcal{A}$:*

  1. *$\mathcal{A}$ sends 1^ℓ, 1^t and $f \in \{0,1\}^\ell$ to the challenger.*
  2. *The challenger generates $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$, $(\mathsf{pp}', \mathsf{ck}) \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$. It flips a coin $b \leftarrow \{0,1\}$ and sends $(\mathsf{pp}', \mathsf{ck})$ to $\mathcal{A}$.*
  3. *In this phase $\mathcal{A}$ can send queries $x \in \{0,1\}^z$ such that $f(x) = 1$ (with no value x queried more than once), to which the challenger replies with $y = \mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x)$, if $b = 0$, or $y \leftarrow \mathbb{Z}_p$, if $b = 1$. $\mathcal{A}$ may also request updated parameters, in which case the challenger samples $g \leftarrow \{0,1\}^\ell$, computes $(\mathsf{pp}', \mathsf{ck}_g) \leftarrow \mathsf{Constrain}_{\mathsf{pp}'}(\mathsf{msk}, g)$ and sends the new parameters pp' to $\mathcal{A}$.*
  4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

  Then $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ for any adversary $\mathcal{A}$.

- **Constraint hiding.** *Consider the following game between a challenger and a stateful PPT adversary $\mathcal{A}$:*

  1. *$\mathcal{A}$ sends 1^ℓ, 1^t and $f^0, f^1 \in \{0,1\}^\ell$ to the challenger.*
  2. *The challenger generates $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$. It flips a coin $b \leftarrow \{0,1\}$ and sends $(\mathsf{pp}', \mathsf{ck}) \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f^b)$ to $\mathcal{A}$.*
  3. *In this phase $\mathcal{A}$ can send queries $x \in \{0,1\}^z$ such that $f^0(x) = f^1(x)$, to which the challenger replies with $y = \mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x)$. $\mathcal{A}$ may also request updated parameters, in which case the challenger samples $g \leftarrow \{0,1\}^\ell$, computes $(\mathsf{pp}', \mathsf{ck}_g) \leftarrow \mathsf{Constrain}_{\mathsf{pp}'}(\mathsf{msk}, g)$ and sends the new parameters pp' to $\mathcal{A}$.*
  4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

  Then $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ for any adversary $\mathcal{A}$.

Our proposed private constrained PRF in Chapter 5 satisfies a stronger form of security, which we define next. In this definition, the adversary can choose the constraints for which the challenger must generate updated parameters. This is a natural strengthening of the previous definition, as in many applications it may not be justified to assume that the distribution of constraints for which the parameters are updated is uniform – it is likely that some constraints are much more probable than others, or it may even be the case that a large portion of the possible constraints is meaningless and has probability zero.

Definition 2.3.2 (Strong security)**.** *A constrained pseudorandom function with updatable parameters* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ *is a* strong single-key selective private constrained pseudorandom function *if the following conditions hold:*

- **Pseudorandomness at constrained points.** *Consider the following game between a challenger and a stateful PPT adversary $\mathcal{A}$:*

1. *$\mathcal{A}$ sends 1^ℓ, 1^t and $f \in \{0,1\}^\ell$ to the challenger.*
2. *The challenger generates $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$, $(\mathsf{pp}', \mathsf{ck}) \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$. It flips a coin $b \leftarrow \{0,1\}$ and sends $(\mathsf{pp}', \mathsf{ck})$ to $\mathcal{A}$.*
3. *In this phase $\mathcal{A}$ can send queries $x \in \{0,1\}^z$ such that $f(x) = 1$ (with no value x queried more than once), to which the challenger replies with $y = \mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x)$, if $b = 0$, or $y \leftarrow \mathbb{Z}_p$, if $b = 1$. $\mathcal{A}$ may also query constraints g, in which case the challenger computes $(\mathsf{pp}', \mathsf{ck}_g) \leftarrow \mathsf{Constrain}_{\mathsf{pp}'}(\mathsf{msk}, g)$ and sends the new parameters pp' to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

Then $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ for any adversary $\mathcal{A}$.

- **Constraint hiding.** *Consider the following game between a challenger and a stateful PPT adversary $\mathcal{A}$:*
  1. *$\mathcal{A}$ sends 1^ℓ, 1^t and $f^0, f^1 \in \{0,1\}^\ell$ to the challenger.*
  2. *The challenger generates $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$. It flips a coin $b \leftarrow \{0,1\}$ and sends $(\mathsf{pp}', \mathsf{ck}) \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f^b)$ to $\mathcal{A}$.*
  3. *In this phase $\mathcal{A}$ can send queries $x \in \{0,1\}^z$ such that $f^0(x) = f^1(x)$, to which the challenger replies with $y = \mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x)$. $\mathcal{A}$ may also query constraints g, in which case the challenger computes $(\mathsf{pp}', \mathsf{ck}_g) \leftarrow \mathsf{Constrain}_{\mathsf{pp}'}(\mathsf{msk}, g)$ and sends the new parameters pp' to $\mathcal{A}$.*
  4. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

  Then $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ for any adversary $\mathcal{A}$.

As the names indicate, strong single-key selective security implies weak single-key selective security, as shown below in Proposition 2.3.3. The converse is not true – in Section 4.5 we show that our first construction of a private constrained PRF satisfies Definition 2.3.1 but not Definition 2.3.2.

Proposition 2.3.3. *Let $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ be a strong single-key selective private constrained pseudorandom function. Then CPRF is a weak single-key selective private constrained pseudorandom function.*

Proof. The proof is straightforward. Let $\mathcal{A}$ be an adversary in the pseudorandomness game of Definition 2.3.1 and consider the following adversary $\mathcal{A}'$ in the pseudorandomness game of Definition 2.3.2. $\mathcal{A}'$ simulates all of the computations of $\mathcal{A}$ and its interaction with the challenger, with the exception that, whenever $\mathcal{A}$ would request an update of the parameters, $\mathcal{A}'$ instead samples $g \leftarrow \{0,1\}^\ell$ by itself and requests updated parameters for g.

It is clear that $\mathcal{A}$ and $\mathcal{A}'$ win their respective security games with the same probability. Since CPRF satisfies strong single-key selective security, $\mathcal{A}'$ has negligible advantage. Therefore any adversary $\mathcal{A}$ against CPRF in the pseudorandomness game of Definition 2.3.1 has negligible advantage. The proof that any adversary has negligible advantage in the constraint-hiding game is analogous. We conclude that CPRF has weak single-key selective security. □

2.4 Applications

Private constrained pseudorandom functions have several applications, including constrained message authentication codes, program watermarking schemes, searchable encryption, deniable encryption [BLW17], as well as functional encryption and reusable garbled circuits [CC17]. In all of these the property of succinct constrained keys is appreciated and the additional need to update the public parameters does not pose a problem. Below we present in detail two applications in which our proposed private CPRFs would be useful.

2.4.1 Database with restricted access

Suppose that an institution wishes to publish an encrypted database and grant each employee (or authorized user) access to certain sections of the database according to their position in the institution hierarchy. Moreover, they want to limit the knowledge that each employee has about their own limits in accessing the database. A possible solution is to employ a private constrained pseudorandom function CPRF in the following way. Consider a symmetric-key encryption scheme SKE with key space equal to the output space of CPRF. For each element x_i of the database $\{x_1, \ldots, x_N\}$, compute an encryption $y_i = \mathsf{SKE.Enc}(\mathsf{CPRF.Eval}(K, i), x_i)$, where K is the master secret key of CPRF. The encrypted database $\{y_1, \ldots, y_N\}$ is made public, while the original database $\{x_1, \ldots, x_N\}$ is kept secret. Additionally, issue for each employee a CPRF constrained key K_f corresponding to the constraint f that accepts an index i if and only if the employee is authorized to access the value x_i.

Now the employees can access the database by interacting with the server which stores it. An employee can send a query Q, to which the server replies with the list I of the indices i that are authorized for the employee (i satisfies f) and such that x_i satisfies Q. Upon receiving I, the employee recovers the relevant data values by computing $x_i = \mathsf{SKE.Dec}(\mathsf{CPRF.ConstrainEval}(K_f, i), y_i)$ for all $i \in I$.

In this example, the constraint-hiding property of CPRF ensures that the employees do not know beforehand which elements of the database they can access – only by interacting with the server can they obtain more information about their limits. Note that the institution has complete freedom on what queries are allowed and they can instruct the server to reject those that are not. If CPRF has succinct keys, then the initial step of distributing constrained keys to all employees becomes much lighter and the employees can store their personal keys more easily. There is a simpler alternative to this protocol, in which the server responds to a query Q by simply sending the set X of elements of the database that satisfy Q and that the employee is authorized to access. However, since the data values may be large, the former approach has the significant advantage that I is generally much smaller than X.

2.4.2 Watermarking

A software watermarking scheme allows one to embed into a program some information, called a mark, which cannot easily be removed without significantly altering the functionality of the program. In its simplest form, a watermarking scheme for a circuit class $\mathcal{C}$ consists of two procedures: Mark, which

receives as input a circuit C and outputs a marked circuit $\tilde{C}$, and Extract, which, given a circuit, outputs either that it is marked or unmarked. The usual requirements are that $\tilde{C}(x) = C(x)$ on all but a negligible fraction of inputs x and that, given the marked circuit $\tilde{C}$, no adversary can produce an unmarked circuit C^* that correctly evaluates C on a significant fraction of inputs. There exist several constructions of watermarking schemes for cryptographic programs, especially PRFs [CHN+16, KW19, YAYX20].

A stronger form of watermarking is message-embedding watermarking, in which the marking procedure receives as additional input a message, and the extraction procedure recovers the embedded message from a marked program. Such a scheme is secure if no adversary can remove the mark or modify the message on a marked program without damaging its functionality.

Quach, Wichs and Zirdelis [QWZ18] proposed a message-embedding watermarking scheme for PRFs which features a private constrained PRF as one of its main components, in addition to regular PRFs and public-key encryption schemes. In this scheme, each marked or unmarked circuit is represented by a key which allows its computation. In the case of a circuit marked with a message μ, this key is of the form $K = (k_\mu, c, s)$, where k_μ is a private CPRF constrained key for a constraint f_μ, c is a ciphertext and s is a PRF key. If we instantiate this scheme with a private CPRF with succinct keys, such as one of the constructions proposed in this thesis, the size of the key k_μ would be independent of the size of the message μ, thus reducing the overall size of the description K of the circuit. Moreover, the watermarking system can easily be adjusted to allow the private CPRF parameters to be updated, as required by our schemes. However, the size of c is linear in the size of μ and therefore, despite this improvement, the size of K remains asymptotically linear in the size of the embedded message.

Chapter 3

The BTVW private constrained PRF

In this chapter we present the main building block of our constructions, which is the private CPRF of Brakerski, Tsabary, Vaikuntanathan, and Wee [BTVW17].[1] Note however that a deep understanding of this private CPRF is not at all necessary in order to understand our schemes. We begin by introducing some background on lattices and the learning with errors problem, followed by two fundamental tools for the private CPRF construction, and finally the description of the private CPRF.

3.1 Lattices and learning with errors

An n-dimensional lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$. This means that every $\mathbf{x} \in \mathcal{L}$ has a neighbourhood U in $\mathbb{R}^n$ such that $U \cap \mathcal{L} = \{\mathbf{x}\}$ and that $\mathbf{0}, \mathbf{x} - \mathbf{y} \in \mathcal{L}$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{L}$. Every n-dimensional lattice is of the form

$$\mathcal{L} = \mathbf{B} \cdot \mathbb{Z}^k = \left\{ \sum_{j=1}^{k} z_j \mathbf{b}_j : z_j \in \mathbb{Z} \right\},$$

where $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ is a set of linearly independent vectors in $\mathbb{R}^n$. Reciprocally, any set of the above form is a lattice. We say that $\mathbf{B}$ is a *basis* of $\mathcal{L}$ and k is the *rank* of $\mathcal{L}$, which is independent from the choice of basis. A lattice is said to be *full-rank* if $k = n$. The *i-th successive minimum* $\lambda_i(\mathcal{L})$, for $i = 1, \ldots, n$, is defined as the minimum radius r such that there exist i linearly independent lattice vectors of norm at most r. In particular, $\lambda_1(\mathcal{L})$ is the length of the shortest non-zero vector of $\mathcal{L}$ and is called the *minimum distance*.

There are many important computational problems on lattices. Here we present two which are particularly relevant in cryptography due to their connection with the learning with errors problem. These are both approximation problems and are parameterized by an approximation factor $\gamma = \gamma(n) \geqslant 1$.

Definition 3.1.1 (GapSVP_γ)**.** *The* decisional approximate shortest vector problem GapSVP_γ *is the following: given a basis* $\mathbf{B}$ *of an* n*-dimensional lattice* $\mathcal{L}$ *such that either* $\lambda_1(\mathcal{L}) \leqslant 1$ *or* $\lambda_1(\mathcal{L}) > \gamma(n)$*, decide which is the case.*

[1] In [BTVW17] two constructions of a private CPRF are proposed. In this thesis we present the first, which has the title "The dual-use technique", and we refer to that one only.

Definition 3.1.2 (SIVP_γ)**.** *The* approximate shortest independent vector problem SIVP_γ *is the following: given a basis* $\mathbf{B}$ *of a full-rank* n*-dimensional lattice* $\mathcal{L}$*, determine a set* $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ *of* n *linearly independent lattice vectors such that* $\|\mathbf{s}_i\|_2 \leqslant \gamma(n)\lambda_n(\mathcal{L})$ *for all* i*.*

The learning with errors problem (LWE) was introduced by Regev [Reg05] and is the basis of many cryptographic constructions. Below we formulate the decision version of LWE.

Definition 3.1.3 (LWE)**.** *Let* n, m, q *be positive integers and* χ *a probability distribution over* $\mathbb{Z}$*. The* decisional learning with errors problem $\mathrm{LWE}_{n,m,q,\chi}$ *is to distinguish the distributions*

$$(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e}) \text{ and } (\mathbf{A}, \mathbf{u}),$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$*,* $\mathbf{s} \leftarrow \mathbb{Z}_q^n$*,* $\mathbf{e} \leftarrow \chi^m$ *and* $\mathbf{u} \leftarrow \mathbb{Z}_q^m$*.*

The following theorem relates learning with errors to worst-case lattice problems for which there are no known efficient classical or quantum algorithms, namely the decisional approximate shortest vector problem, GapSVP_γ, and the approximate shortest independent vector problem, SIVP_γ. This provides the basis for the hardness of LWE to be accepted as a standard assumption in cryptography.

Theorem 3.1.4 ([Reg05, Pei09])**.** *Let* $n \in \mathbb{N}$*,* $m = \mathrm{poly}(n)$*,* $q \leqslant 2^{\mathrm{poly}(n)}$*, and let* χ *be the discrete Gaussian distribution with parameter* $\alpha\, q$*, where* $2\sqrt{n}/q \leqslant \alpha < 1$*. If there is an efficient algorithm that solves* $\mathrm{LWE}_{n,m,q,\chi}$*, then:*

- *There is an efficient quantum algorithm that solves* GapSVP_γ *and* SIVP_γ *on any* n*-dimensional lattice, for some* $\gamma = \tilde{\mathcal{O}}(n/\alpha)$*.*
- *If in addition* $q \geqslant \tilde{\mathcal{O}}(2^{n/2})$*, there is an efficient classical algorithm that solves* GapSVP_γ *on any* n*-dimensional lattice, for some* $\gamma = \tilde{\mathcal{O}}(n/\alpha)$*.*

A classical reduction from LWE to GapSVP_γ also holds for polynomial moduli q [BLP+13]. For more information about computational problems on lattices and a general survey of lattice cryptography, see [Pei16].

3.2 Homomorphic evaluation over matrices

Given positive integers n, q, we define the *gadget matrix* [MP12]

$$\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} = \mathsf{diag}(\mathbf{g}, \ldots, \mathbf{g}) \in \mathbb{Z}_q^{n\times n\lceil \log q\rceil},$$

where $\mathbf{g} = (1, 2, 4, \ldots, 2^{\lceil \log q\rceil - 1})$ and $\mathbf{I}_n \in \mathbb{Z}_q^{n\times n}$ is the identity matrix. We denote by $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0,1\}^{n\lceil \log q\rceil}$ the function such that the i-th block of size $\lceil \log q\rceil$ of $\mathbf{G}^{-1}(\mathbf{u})$ consists of the bits of the binary decomposition of the i-th entry of $\mathbf{u}$, and we extend $\mathbf{G}^{-1}$ to matrices by applying it column-wise. Then, for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times k}$, we have $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

We present in the following theorem the properties of the algorithms for evaluation of circuits over matrices used in [BTVW17], which are based on the algorithms developed in [GSW13, BGG+14].

Theorem 3.2.1. *There exist efficient deterministic algorithms* EvalF *and* EvalFX *such that for all* $n, q, z \in \mathbb{N}$*, for any matrices* $\mathbf{A}_1, \ldots, \mathbf{A}_z \in \mathbb{Z}_q^{n \times n \log q}$*, for any Boolean circuit* $f : \{0,1\}^z \to \{0,1\}$ *of depth* t *and for any* $x = (x_1, \ldots, x_z) \in \{0,1\}^z$*, the following properties hold.*

- *The outputs* $\mathbf{H}_f = \mathsf{EvalF}(f, \mathbf{A}_1, \ldots, \mathbf{A}_z)$ *and* $\mathbf{H}_{f,x} = \mathsf{EvalFX}(f, x, \mathbf{A}_1, \ldots, \mathbf{A}_z)$ *are matrices in* $\mathbb{Z}^{zn \log q \times n \log q}$.
- $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,x}\|_\infty \leqslant (n \log q)^{\mathcal{O}(t)}$.
- $[\mathbf{A}_1 - x_1\mathbf{G} \mid \ldots \mid \mathbf{A}_z - x_z\mathbf{G}] \cdot \mathbf{H}_{f,x} = [\mathbf{A}_1 \mid \ldots \mid \mathbf{A}_z] \cdot \mathbf{H}_f - f(x)\mathbf{G} \pmod q$.

3.3 Fully homomorphic encryption

A fully homomorphic encryption (FHE) scheme is a secure encryption scheme with an additional algorithm Eval, which receives as input a function f and a ciphertext ct encrypting a plaintext x and computes a ciphertext encrypting $f(x)$. It has the following syntax:

- $\mathsf{KeyGen}(1^\lambda)$ receives as input the security parameter λ and outputs a secret key msk.
- $\mathsf{Enc}(\mathsf{msk}, x)$ receives as input the key msk and a message x, and outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{msk}, c)$ receives as input the key msk and a ciphertext c. It outputs either a message x.
- $\mathsf{Eval}(f, c)$ receives as input a function f and a ciphertext c. It outputs another ciphertext c'.

Correctness. For any message x, function f and $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$,

$$\mathsf{Dec}(\mathsf{msk}, \mathsf{Eval}(f, \mathsf{Enc}(\mathsf{msk}, x))) = f(x).$$

The usual security requirement for FHE is IND-CPA security (see Section 5.1). However, we will not directly analyze the security of any FHE system. Now we describe the variant of the FHE scheme of Gorbunov, Vaikuntanathan and Wee [GSW13] that appears in the private CPRF scheme of Brakerski et al. [BTVW17]. We present this scheme because in the next section some of its algorithms will be used explicitly and others implicitly. The key generation, encryption, and decryption algorithms are as follows:

- $\mathsf{KeyGen}(1^\lambda)$: Let $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and output $\mathbf{v}^\mathsf{T} = [\mathbf{s}^\mathsf{T} \mid -1]$ as the secret key.
- $\mathsf{Enc}(\mathbf{v}, \mu \in \{0,1\})$: Write $\mathbf{v}^\mathsf{T} = [\mathbf{s}^\mathsf{T} \mid -1]$. Sample $\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{R} \leftarrow \{0,1\}^{m \times m}$ and $\mathbf{e} \leftarrow \chi^m$. Output the ciphertext
$$\Psi = \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T} \end{pmatrix} \mathbf{R} + \mu\mathbf{G}.$$
- $\mathsf{Dec}(\mathbf{v}, \Psi)$: Compute the inner product $y = \langle \mathbf{v}, \mathbf{c} \rangle$, where $\mathbf{c}$ is the penultimate column of Ψ, and verify whether y is closer to 0 or to $q - 2^{\log q - 1}$ modulo q. In the former case output $\mu = 0$ and in the latter output $\mu = 1$.

The homomorphic evaluation algorithm Eval is defined by the basic operations of addition and multiplication. The homomorphic addition of two ciphertexts Ψ_1 and Ψ_2 is defined as $\Psi_1 + \Psi_2$ and their homomorphic multiplication as $\Psi_1 \cdot \mathbf{G}^{-1}(\Psi_2)$. Observe that, when $\mathbf{v}$ and Ψ are generated as above, $[\mathbf{s}^\mathsf{T} \,|\, -1]\Psi = -\mathbf{e}^\mathsf{T}\mathbf{R} + \mu[\mathbf{s}^\mathsf{T} \,|\, -1]\mathbf{G} \approx \mu[\mathbf{s}^\mathsf{T} \,|\, -1]\mathbf{G}$, which implies that in the decryption algorithm $y \approx -\mu\, 2^{\log q - 1}$, justifying its correctness.

3.4 LWE-based private constrained PRF

Let $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ be the GSW fully homomorphic encryption scheme [GSW13] and let EvalF, EvalFX be the algorithms of Theorem 3.2.1. The private constrained PRF scheme BTVW of Brakerski, Tsabary, Vaikuntanathan, and Wee [BTVW17], supporting constraints computable by circuits of polynomial size and bounded depth, consists of the following algorithms. We follow the presentation of the original, except for the algorithm BTVW.Constrain, where we identify two subroutines which will be used separately in our construction in the next chapter.

- $\mathsf{BTVW.KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$: The input parameters are the security parameter λ, the maximum description length ℓ of constraint functions, their input length z and their maximum depth t.

  Let $L = \ell\,(n+1)^2 \log^2 q$. Sample $\mathbf{B}, \mathbf{B}_1, \ldots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times (n+1)\log q}$, as well as $\mathbf{C}_0, \mathbf{C}_1, \mathbf{D} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{s} \leftarrow \mathbb{Z}_q^n$. Output

  $$\mathsf{msk} = \mathbf{s}, \;\; \mathsf{pp} = \left(\mathbf{B}, \{\mathbf{B}_j\}_{j\in[L]}, \mathbf{C}_0, \mathbf{C}_1, \mathbf{D}\right)$$

  as the master secret key and public parameters, respectively.

- $\mathsf{BTVW.Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$: Let $\mathcal{U}_x : \{0,1\}^\ell \to \{0,1\}$ be the universal circuit that takes as input the description of a function f and outputs $f(x)$. Consider the circuit $\widehat{\mathcal{U}}_x : \{0,1\}^L \to \mathbb{Z}_q^{n\times(n+1)\log q}$, which takes as input a FHE encryption Ψ of the description of a function f and outputs $\overline{\Psi}_x$, which is the matrix obtained from $\Psi_x = \mathsf{FHE.Eval}(\mathcal{U}_x, \Psi)$ by removing the last row. Let also $\mathcal{T}_x(y_0, y_1)$ be the circuit that computes the product $\prod_{k\in[z]} y_{x_k}$ through a balanced binary multiplication tree. Compute

  $$\begin{aligned}
  \mathbf{H}_{\widehat{\mathcal{U}}_x} &= \mathsf{EvalF}\left(\widehat{\mathcal{U}}_x, \mathbf{B}_1, \ldots, \mathbf{B}_L\right), \\
  \mathbf{B}_{\widehat{\mathcal{U}}_x} &= [\mathbf{B}_1 \,|\, \ldots \,|\, \mathbf{B}_L] \cdot \mathbf{H}_{\widehat{\mathcal{U}}_x}, \\
  \mathbf{C}_x &= \mathsf{EvalF}\left(\mathcal{T}_x, \mathbf{C}_0, \mathbf{C}_1\right), \\
  \mathbf{M}_x &= \mathbf{D}\mathbf{G}^{-1}(\mathbf{C}_x)
  \end{aligned}$$

  and output

  $$y = \left\lfloor \mathbf{s}^\mathsf{T}\mathbf{B}_{\widehat{\mathcal{U}}_x}\mathbf{G}^{-1}(\mathbf{M}_x) \right\rceil_p,$$

  where the rounding operation $\lfloor\cdot\rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$, defined for $p < q$ by $\lfloor k \rceil_p = \lfloor kp/q \rceil$, is applied coordinate-wise.

- $\mathsf{BTVW.Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$: Sample $\mathbf{e}_0 \leftarrow \chi^{(n+1)\log q}$. Consider the following two subroutines, both of which receive $\beta \in \{0,1\}$ as one their inputs.

  - $\mathsf{BTVW.GenP}_{\mathsf{pp}}(\mathsf{msk}, \mathbf{e}_0, \beta)$: Sample $\mathbf{R} \leftarrow \{0,1\}^{(n+1)\log q \times (n+1)\log q}$. Output

$$\Psi = \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}_0^\mathsf{T} \end{pmatrix} \mathbf{R} + \beta \mathbf{G}.$$

  - $\mathsf{BTVW.GenC}_{\mathsf{pp}}(\mathsf{msk}, \mathbf{A}, \beta)$: Sample $\mathbf{e} \leftarrow \chi^{(n+1)\log q}$. Output

$$\mathbf{c}^\mathsf{T} = \mathbf{s}^\mathsf{T}(\mathbf{A} - \beta\overline{\mathbf{G}}) + \mathbf{e}^\mathsf{T},$$

    where $\overline{\mathbf{G}}$ denotes the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\log q}$ with its last row removed.

  Let $(f_1, \ldots, f_\ell)$ be the description of f. Generate $\Psi_i \leftarrow \mathsf{BTVW.GenP}_{\mathsf{pp}}(\mathsf{msk}, \mathbf{e}_0, f_i)$ for all $i \in [\ell]$ and let $(\psi_1, \ldots, \psi_L)$ be the binary representation of $\Psi = [\Psi_1 \mid \ldots \mid \Psi_\ell]$. For each $j \in L$, generate $\mathbf{c}_j \leftarrow \mathsf{BTVW.GenC}_{\mathsf{pp}}(\mathsf{msk}, \mathbf{B}_j, \psi_j)$.

  Output $\mathsf{ck} = (\Psi, \{\mathbf{c}_j\}_{j\in[L]})$ as the constrained key.

- $\mathsf{BTVW.ConstrainEval}_{\mathsf{pp}}(\mathsf{ck}, x)$: Parse $\mathsf{ck} = (\Psi, \{\mathbf{c}_j\}_{j\in[L]})$. and define $\mathcal{U}_x, \widehat{\mathcal{U}}_x, \mathbf{M}_x$ as in the algorithm BTVW.Eval above. Compute:

$$\begin{aligned} \Psi_x &= \mathsf{FHE.Eval}(\mathcal{U}_x, \Psi), \\ \mathbf{H}_{\widehat{\mathcal{U}}_x, \Psi} &= \mathsf{EvalFX}\left(\widehat{\mathcal{U}}_x, \Psi, \mathbf{B}_1, \ldots, \mathbf{B}_L\right), \\ \mathbf{c}_{\widehat{\mathcal{U}}_x} &= [\mathbf{c}_1 \mid \ldots \mid \mathbf{c}_L] \cdot \mathbf{H}_{\widehat{\mathcal{U}}_x, \Psi} + \underline{\Psi}_x. \end{aligned}$$

  Output

$$y' = \left\lfloor \mathbf{c}_{\widehat{\mathcal{U}}_x} \mathbf{G}^{-1}(\mathbf{M}_x) \right\rceil_p.$$

Theorem 3.4.1 ([BTVW17])**.** *Under the LWE assumption,* BTVW *is a single-key selective private constrained pseudorandom function (as per Definition 2.2.2).*

Chapter 4

Succinct-key private constrained PRF from attribute-based encryption

In this chapter we present our construction of a private constrained PRF scheme with succinct constrained keys. This scheme is based on the private constrained PRF of Brakerski, Tsabary, Vaikuntanathan, and Wee [BTVW17]. Our approach to reduce the size of the constrained keys using attribute-based encryption is inspired by the techniques used by Brakerski and Vaikuntanathan [BV15] to adapt their constrained PRF in order to have succinct keys.

4.1 Attribute-based encryption

Attribute-based encryption (ABE), first introduced by Sahai and Waters [SW05], is a generalization of public-key encryption that allows multiple users to have different permissions to access encrypted data. In the case of key-policy ABE, ciphertexts are associated to attributes and personal keys are associated to constraints over the attribute space. A key corresponding to a constraint f is allowed to decrypt a ciphertext with attribute x if and only if $f(x) = 0$.[1] There also exists ciphertext-policy ABE, in which ciphertexts are associated to constraints and keys to attributes, but only the key-policy variant will be considered in this work. There is an evident similarity between ABE and CPRFs – in a CPRF there are constrained keys that control the access to PRF values, while in ABE constrained keys control the ability to decrypt ciphertexts.

A (key-policy) attribute-based encryption scheme is a tuple $(\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Enc}, \mathsf{Dec})$ of PPT algorithms with the following syntax:

- $\mathsf{Setup}(1^\lambda)$ receives as input the security parameter λ and outputs a secret key msk and a public key pk.
- $\mathsf{Constrain}(\mathsf{msk}, f)$ receives as input the secret key msk and a constraint f. It outputs a constrained key ck.

[1] As in the context of CPRFs, we interpret $f(x) = 0$ as f being satisfied by x.

- $\mathsf{Enc}(\mathsf{pk}, x, \mu)$ receives as input the public key pk, an attribute x and a message μ, and outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{ck}, c)$ receives as input a constrained key ck and a ciphertext c. It outputs either a message μ or the symbol $\perp$.

Remark. The second algorithm in the definition of ABE is usually labelled *KeyGen*, but we renamed it to *Constrain* to avoid confusion regarding the parallelism between ABE and CPRFs (in the usual nomenclature, *Setup* in ABE is analogous to *KeyGen* in CPRFs, but *KeyGen* in ABE is analogous to *Constrain* in CPRFs).

Correctness. For any attribute x, constraint f such that $f(x) = 0$, and message μ,

$$\Pr\left[\mathsf{Dec}(\mathsf{ck}, c) = \mu\right] \geqslant 1 - \mathrm{negl}(\lambda),$$

where $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x, \mu)$ and $\mathsf{ck} \leftarrow \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$.

We consider the notion of selective security for ABE in the multi-message setting, which we define next. This is equivalent to selective security for a single message.

Definition 4.1.1 (Selective security)**.** *Let* $(\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Enc}, \mathsf{Dec})$ *be an attribute-based encryption scheme and consider the following game between a challenger and a stateful PPT adversary* $\mathcal{A}$*:*

1. $\mathcal{A}$ *sends attributes* $x_1, \ldots, x_k$ *to the challenger.*
2. *The challenger generates* $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, flips a coin* $b \leftarrow \{0,1\}$ *and sends* pk *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *can query predicates* f *such that* $f(x_i) = 1$ *for all* $i \in [k]$*, to which the challenger replies with* $\mathsf{Constrain}(\mathsf{msk}, f)$*. The adversary sends pairs of messages* $(\mu_{0,1}, \mu_{1,1}), \ldots, (\mu_{0,k}, \mu_{1,k})$ *to the challenger.*
4. *The challenger sends* $\{c_i\}_{i\in[k]}$ *to* $\mathcal{A}$*, where* $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i, \mu_{b,i})$*.*
5. *At this step* $\mathcal{A}$ *is allowed to make queries as in step 3. Finally, it outputs a guess* $b' \in \{0,1\}$*.*

We say that the ABE scheme is selectively secure *if* $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$ *for any PPT adversary* $\mathcal{A}$*.*

For our construction we will need an ABE scheme with short constrained keys, such as the lattice-based scheme of Boneh et al. [BGG+14].

Theorem 4.1.2 ([BGG+14])**.** *Under the LWE assumption, there exists a selectively secure ABE scheme for the class of depth-*t *circuits in which the constrained keys have size* $\mathrm{poly}(\lambda, t)$*, where* λ *is the security parameter.*

4.2 Succinct-key private constrained PRF

The starting point for this construction is the private CPRF of Brakerski, Tsabary, Vaikuntanathan, and Wee [BTVW17], in which the first component of a constrained key is of the form $(\Psi_1, \ldots, \Psi_\ell)$ and each

Ψ_i only depends on the i-th bit f_i of the description of the constraint f. Relying on this fact, we encrypt with ABE each Ψ_i for both the cases $f_i = 0$ and $f_i = 1$ and place those encryptions in the public parameters. Then we give as the constrained key for our scheme an ABE constrained key that allows the user to decrypt only the relevant ciphertexts in the parameters. A similar procedure applies to the second component of the key. This technique has been used before by Brakerski and Vaikuntanathan [BV15] to turn their construction of a standard CPRF into a CPRF with succinct keys. Additionally, we permutate the aforementioned ciphertexts using bits $d_1, \ldots, d_\ell$ in order to keep the constraint hidden. The additional information contained in the updated parameters allows the user to decrypt the correct ciphertexts, despite this permutation.

Let $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.Constrain}, \mathsf{ABE.Dec})$ be an attribute-based encryption scheme and let $\mathsf{BTVW} = (\mathsf{BTVW.KeyGen}, \mathsf{BTVW.Eval}, \mathsf{BTVW.Constrain}, \mathsf{BTVW.ConstrainEval})$ be the private constrained pseudorandom function scheme of [BTVW17]. Our succinct-key private constrained PRF with updatable parameters SCPRF consists of the following algorithms.

- $\mathsf{SCPRF.KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$: The input parameters are the security parameter λ, the maximum description length ℓ of constraint functions, their input length z and their maximum depth t.

  Let $(\mathsf{BTVW.msk}, \mathsf{BTVW.pp}) \leftarrow \mathsf{BTVW.KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$. Sample $\mathbf{e}_0$ and $\mathbf{B}_1, \ldots, \mathbf{B}_L$ as in the BTVW CPRF and then generate

  $$\Psi_{i,\beta} \leftarrow \mathsf{BTVW.GenP}_{\mathsf{BTVW.pp}}(\mathsf{BTVW.msk}, \mathbf{e}_0, \beta),$$
  $$\mathbf{c}_{j,\beta} \leftarrow \mathsf{BTVW.GenC}_{\mathsf{BTVW.pp}}(\mathsf{BTVW.msk}, \mathbf{B}_j, \beta)$$

  for all $i \in [\ell], j \in [L]$ and $\beta \in \{0,1\}$, where the parameter L is set as in $\mathsf{BTVW.KeyGen}$.

  Set up independently two ABE schemes as follows: $(\mathsf{ABE.msk}_1, \mathsf{ABE.pk}_1) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and $(\mathsf{ABE.msk}_2, \mathsf{ABE.pk}_2) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$. Then sample $d_1, \ldots, d_\ell \leftarrow \{0,1\}$ and compute

  $$\mathbf{a}_{i,\beta} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1, (i,\beta), \Psi_{i,\beta \oplus d_i}),$$
  $$\mathbf{b}_{j,\beta} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2, (j,\beta), \mathbf{c}_{j,\beta})$$

  for all $i \in [\ell], j \in [L], \beta \in \{0,1\}$. Output

  $$\mathsf{msk} = \left(\mathsf{BTVW.msk}, \{d_i\}_{i\in[\ell]}, \mathsf{ABE.msk}_1, \mathsf{ABE.msk}_2\right),$$
  $$\mathsf{pp} = \left(\mathsf{BTVW.pp}, \{\mathbf{a}_{i,\beta}\}_{i\in[\ell],\beta\in\{0,1\}}, \{\mathbf{b}_{j,\beta}\}_{j\in[L],\beta\in\{0,1\}}, \mathsf{ABE.pk}_1, \mathsf{ABE.pk}_2\right)$$

  as the master secret key and the public parameters, respectively.

- $\mathsf{SCPRF.Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$: Output $y = \mathsf{BTVW.Eval}_{\mathsf{BTVW.pp}}(\mathsf{BTVW.msk}, x)$.

- $\mathsf{SCPRF.Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$: Let $(f_1, \ldots, f_\ell)$ be the description of f and $u = (u_1, \ldots, u_\ell) = (f_1 \oplus d_1, \ldots, f_\ell \oplus d_\ell)$. Compute $\Psi_i \leftarrow \mathsf{ABE.Dec}(\mathsf{ABE.msk}_1, \mathbf{a}_{i,u_i})$ for all $i \in [\ell]$. Let $\psi_1, \ldots, \psi_L$ be the digits

of the binary decomposition of $\Psi = [\Psi_1 \mid \ldots \mid \Psi_\ell]$ and define the predicates

$$\Phi_u(i, \beta) = \begin{cases} 0, & \text{if } \beta = u_i \\ 1, & \text{otherwise,} \end{cases}$$

$$\Lambda_\Psi(j, \beta) = \begin{cases} 0, & \text{if } \beta = \psi_j \\ 1, & \text{otherwise.} \end{cases}$$

Let $\mathsf{ABE.ck}_1 \leftarrow \mathsf{ABE.Constrain}(\mathsf{ABE.msk}_1, \Phi_u)$ and $\mathsf{ABE.ck}_2 \leftarrow \mathsf{ABE.Constrain}(\mathsf{ABE.msk}_2, \Lambda_\Psi)$. Output $\mathsf{ck} = (\mathsf{ABE.ck}_1, \mathsf{ABE.ck}_2)$ as the constrained key and $\mathsf{pp}' = (\mathsf{pp}, u)$ as the new public parameters.

- $\mathsf{SCPRF.ConstrainEval}_{\mathsf{pp}'}(\mathsf{ck}, x)$: Retrieve u from the public parameters. For each $i \in [\ell]$, compute $\Psi_i = \mathsf{ABE.Dec}(\mathsf{ABE.ck}_1, \Phi_u, \mathbf{a}_{i,u_i})$. Observe that the ABE constrained key for Φ_u allows the decryption of $\mathbf{a}_{i,u_i}$ but not $\mathbf{a}_{i,1-u_i}$. Let $(\psi_1, \ldots, \psi_L)$ be the binary decomposition of $\Psi = [\Psi_1 | \ldots | \Psi_\ell]$. Compute $\mathbf{c}_j = \mathsf{ABE.Dec}(\mathsf{ABE.ck}_2, \Lambda_\Psi, \mathbf{b}_{j,\psi_j})$ for all $j \in [L]$.

  Output $y = \mathsf{BTVW.ConstrainEval}_{\mathsf{BTVW.pp}}((\Psi, \{\mathbf{c}_j\}_{j\in[L]}), x)$.

4.3 Proof of security

In this section we show that the construction presented above is correct and weakly secure, as per Definition 2.3.1. Note that it supports the same class of constraints as BTVW, which is the set of predicates computable by circuits of polynomial size and bounded depth.

Theorem 4.3.1 (Correctness)**.** *Suppose* ABE *is a correct attribute-based encryption scheme and* BTVW *is correct. Then* SCPRF *is also correct.*

Proof. Correctness of our scheme follows from the correctness of the ABE scheme and the BTVW constrained PRF scheme. It is clear that $\mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x) = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$ for any $x \in \{0,1\}^z$, since Eval depends only on the parameters BTVW.pp. Let ck be a constrained key for a predicate f. By the correctness of ABE, the decryption of $\mathbf{a}_{i,u_i}$ yields $\Psi_i = \Psi_{i,f_i}$ for all $i \in [\ell]$. Analogously, $\mathbf{c}_j = \mathbf{c}_{j,\psi_j}$ for all $j \in [L]$, where $(\psi_1, \ldots, \psi_L)$ is the binary decomposition of $\Psi = [\Psi_1 | \ldots | \Psi_\ell]$. Therefore $(\Psi, \{\mathbf{c}_j\})$ is a BTVW constrained key for f. For any $x \in \{0,1\}^z$, we have

$$\mathsf{SCPRF.Eval}_{\mathsf{pp}}(\mathsf{msk}, x) = \mathsf{SCPRF.ConstrainEval}_{\mathsf{pp}'}(\mathsf{ck}, x)$$

if and only if

$$\mathsf{BTVW.Eval}_{\mathsf{BTVW.pp}}(\mathsf{BTVW.msk}, x) = \mathsf{BTVW.ConstrainEval}_{\mathsf{BTVW.pp}}((\Psi, \{\mathbf{c}_j\}_{j\in[L]}), x).$$

Since BTVW is a correct constrained PRF, if $f(x) = 0$ then the above equations hold with all but negligible probability.

□

Theorem 4.3.2 (Pseudorandomness)**.** *Suppose* ABE *is a selectively secure attribute-based encryption scheme and* BTVW *is pseudorandom at constrained points, as per Definition 2.2.2. Then* SCPRF *satisfies the pseudorandomness at constrained points property of Definition 2.3.1.*

Proof. We divide the proof into two parts. In the first we show that, under the assumption of selective security of the ABE scheme, the following security games are computationally indistinguishable.

Game 0. This is the original pseudorandomness security game of Definition 2.3.1 for our construction.

Game 1. This game is identical to the previous one, only with a change in the way in which the ciphertexts $\{\mathbf{a}_{i,\beta}\}$ are generated. The challenger sets $\mathbf{a}_{i,1-u_i} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1,(i,1-u_i),\mathbf{0})$ for all $i \in [\ell]$, where $(u_1,\ldots,u_\ell) = (f_1 \oplus d_1,\ldots,f_\ell \oplus d_\ell)$ and $\mathbf{0}$ represents the zero vector of appropriate size. The remaining ciphertexts, namely those of the form $\mathbf{a}_{i,u_i}$, are computed as usual.

Game 2. In this game the challenger computes $\mathbf{b}_{j,1-\psi_j} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2,(j,1-\psi_j),\mathbf{0})$ for all $j \in [L]$ before sending these ciphertexts to the adversary as part of the public parameters. Game 2 is otherwise identical to Game 1, with the ciphertexts of the form $\mathbf{b}_{j,\psi_j}$ generated as usual.

Claim 4.3.2.1. *If* ABE *is a selectively secure attribute-based encryption scheme, then Game 0 and Game 1 are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT adversary in the experiment of distinguishing Game 0 from Game 1. At the start of this experiment the challenger randomly chooses $\beta \leftarrow \{0,1\}$. The challenger and the adversary then play Game 0, if $\beta = 0$, or Game 1, if $\beta = 1$. Finally, $\mathcal{A}$ outputs a guess β' and wins if $\beta' = \beta$.

Given $\mathcal{A}$ we construct a PPT adversary $\mathcal{B}$ against ABE in the selective security game of Definition 4.1.1 as follows. $\mathcal{B}$ begins by running $\mathcal{A}$, which replies with a constraint $f \in \{0,1\}^\ell$. Then it generates the public parameters $(\mathsf{BTVW.pp},\{\mathbf{b}_{j,\beta}\},\mathsf{ABE.pk}_2)$, the secret key components $(\mathsf{BTVW.msk},\{d_i\},\mathsf{ABE.msk}_2)$ and computes $\{\Psi_{i,\beta}\}$ as in the SCPRF key generation algorithm. It also chooses $b \leftarrow \{0,1\}$ uniformly.

The adversary $\mathcal{B}$ computes $(u_1,\ldots,u_\ell) = (f_1 \oplus d_1,\ldots,f_\ell \oplus d_\ell)$ and sends to the ABE challenger the attribute sequence $\{(i,1-u_i)\}_{i\in[\ell]}$. After the challenger generates $(\mathsf{ABE.msk}_1,\mathsf{ABE.pk}_1) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and samples $\beta \leftarrow \{0,1\}$, $\mathcal{B}$ receives $\mathsf{ABE.pk}_1$ and it sends back the message pairs $\{(\mu_{0,i},\mu_{1,i})\}_{i\in[\ell]}$ given by $\mu_{0,i} = \Psi_{i,1-f_i}$ and $\mu_{1,i} = \mathbf{0}$. It then receives ciphertexts $\mathbf{a}_{i,1-u_i}$ which encrypt either $\Psi_{i,1-f_i}$ or $\mathbf{0}$, depending on the value of β.

Using the public key provided by the challenger, $\mathcal{B}$ computes $\mathbf{a}_{i,u_i} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1,(i,u_i),\Psi_{i,f_i})$. Then it queries the ABE challenger on the constraint Φ_u. Since $\Phi_u(i,1-u_i) = 1$ for all i, the query is allowed and the challenger must reply with a constrained key $\mathsf{ABE.ck}_1$. $\mathcal{B}$ generates $\mathsf{ABE.ck}_2$ as in the SCPRF.Constrain algorithm and sends updated public parameters $\mathsf{pp}' = (\mathsf{BTVW.pp},\{\mathbf{a}_{i,\beta}\},\{\mathbf{b}_{j,\beta}\},\mathsf{ABE.pk}_1,\mathsf{ABE.pk}_2,u)$ and a constrained key $\mathsf{ck} = (\mathsf{ABE.ck}_1,\mathsf{ABE.ck}_2)$ to $\mathcal{A}$.

If $\mathcal{A}$ issues a point query x, $\mathcal{B}$ uses $\mathsf{BTVW.msk}$ to compute and send the corresponding value y, which is $y = \mathsf{BTVW.Eval}_{\mathsf{BTVW.pp}}(\mathsf{BTVW.msk},x)$, if $b = 0$, or $y \leftarrow \mathbb{Z}_p$, if $b = 1$. Whenever $\mathcal{A}$ requests new parameters, $\mathcal{B}$ samples $g \leftarrow \{0,1\}^\ell$, computes $u' = (g_1 \oplus d_1,\ldots,g_\ell \oplus d_\ell)$ and replies with (pp',u'). Once $\mathcal{A}$ outputs its guess β', $\mathcal{B}$ also outputs β'.

The game that $\mathcal{A}$ plays has the distribution of Game 0 if $\beta = 0$ and the distribution of Game 1 if $\beta = 1$. It follows that $\mathcal{B}$ wins against the ABE challenger with the same probability that $\mathcal{A}$ successfully distinguishes the two games. By selective security of ABE, $\mathcal{B}$ has negligible advantage, hence $\mathcal{A}$ has negligible advantage as well. ■

Claim 4.3.2.2. *If* ABE *is a selectively secure attribute-based encryption scheme, then Game 1 and Game 2 are computationally indistinguishable.*

The proof of Claim 4.3.2.2 is analogous to the proof of Claim 4.3.2.1, with the adversary $\mathcal{B}$ choosing the attributes (j, ψ_j) and the messages $\mu_{0,j} = \mathbf{c}_{j,1-\psi_j}$, $\mu_{1,j} = \mathbf{0}$. It follows that Games 0 and 2 are also indistinguishable under our security assumption for ABE. We now show that $\mathcal{A}$ has negligible advantage in Game 2.

Claim 4.3.2.3. *If the constrained PRF* BTVW *is pseudorandom, then the advantage of any PPT adversary in Game 2 is negligible.*

Proof. We show that, for any computational adversary $\mathcal{A}$ in Game 2, there exists an adversary $\mathcal{B}$ that wins the pseudorandomness security game of the BTVW scheme with the same probability as $\mathcal{A}$. We define $\mathcal{B}$ as follows.

At the beginning of the game, the adversary $\mathcal{B}$ runs $\mathcal{A}$ and receives a constraint $f \in \{0,1\}^\ell$, which it forwards to the challenger. In the next step, $\mathcal{B}$ receives BTVW public parameters BTVW.pp and a constrained key $([\Psi_1 \mid \ldots \mid \Psi_\ell], \{\mathbf{c}_j\})$ from the challenger. Then it generates $(\mathsf{ABE.msk}_1, \mathsf{ABE.pk}_1) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, $(\mathsf{ABE.msk}_2, \mathsf{ABE.pk}_2) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, samples $d_1, \ldots, d_\ell \leftarrow \{0,1\}$ and computes the ciphertexts

$$\begin{aligned}
\mathbf{a}_{i,u_i} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1, (i, u_i), \Psi_i),\\
\mathbf{a}_{i,1-u_i} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1, (i, 1-u_i), \mathbf{0}),\\
\mathbf{b}_{j,\psi_j} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2, (j, \psi_j), \mathbf{c}_j),\\
\mathbf{b}_{j,1-\psi_j} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2, (j, 1-\psi_j), \mathbf{0}),
\end{aligned}$$

where $(u_1, \ldots, u_\ell) = (f_1 \oplus d_1, \ldots, f_\ell \oplus d_\ell)$ and $(\psi_1, \ldots, \psi_L)$ is the binary representation of the matrix $\Psi = [\Psi_1 \mid \ldots \mid \Psi_\ell]$. Additionally, $\mathcal{B}$ generates $\mathsf{ABE.ck}_1 \leftarrow \mathsf{ABE.Constrain}(\mathsf{ABE.msk}_1, \Phi_u)$ and $\mathsf{ABE.ck}_2 \leftarrow \mathsf{ABE.Constrain}(\mathsf{ABE.msk}_2, \Lambda_\Psi)$ as in the algorithm SCPRF.Constrain. Finally, $\mathcal{B}$ sends to $\mathcal{A}$ the updated public parameters $\mathsf{pp}' = (\mathsf{BTVW.pp}, \{\mathbf{a}_{i,\beta}\}, \{\mathbf{b}_{j,\beta}\}, \mathsf{ABE.pk}_1, \mathsf{ABE.pk}_2, u)$ and the constrained key $(\mathsf{ABE.ck}_1, \mathsf{ABE.ck}_2)$.

In order to obtain the final guess of $\mathcal{A}$, $\mathcal{B}$ must be able to reply to its queries. Whenever $\mathcal{A}$ queries a point $x \in \{0,1\}^z$, $\mathcal{B}$ makes that same query to the BTVW challenger. The challenger responds with a value y, which $\mathcal{B}$ forwards to $\mathcal{A}$ as its reply. If the adversary $\mathcal{A}$ asks for an update of the public parameters, $\mathcal{B}$ samples $g \leftarrow \{0,1\}^\ell$, computes $u' = (g_1 \oplus d_1, \ldots, g_\ell \oplus d_\ell)$ and sends (pp', u'). Once $\mathcal{A}$ halts and outputs a guess b', $\mathcal{B}$ outputs b' as well.

Observe that $\mathcal{B}$ wins the pseudorandomness game against the BTVW protocol if and only if $\mathcal{A}$ wins the game to which $\mathcal{B}$ challenged it, and that this game has the correct distribution of Game 2. By the pseudorandomness property of the BTVW protocol, $\mathcal{B}$ has negligible advantage, hence $\mathcal{A}$ has negligible advantage in Game 2. ■

Since Game 0 is computationally indistinguishable from Game 2, we conclude that any PPT adversary has negligible advantage in Game 0. Therefore SCPRF is pseudorandom.

□

Theorem 4.3.3 (Constraint hiding)**.** *Suppose* ABE *is a selectively secure attribute-based encryption scheme and* BTVW *is constraint-hiding, as per Definition 2.2.2. Then* SCPRF *satisfies the constraint-hiding property of Definition 2.3.1.*

Proof. We show that the following security games are computationally indistinguishable.

Game 0. This is the constraint hiding security game of Definition 2.3.1 for the SCPRF protocol. The game begins with the adversary $\mathcal{A}$ submitting two constraints f^0, f^1. Then the challenger samples $b \leftarrow \{0,1\}$ and sends to $\mathcal{A}$ updated public parameters pp' and a constrained key ck, generated for f^b according to the algorithms SCPRF.KeyGen and SCPRF.Constrain. $\mathcal{A}$ performs additional computations, with the possibility of sending queries x such that $f^0(x) = f^1(x)$, to which the challenger responds with the value of SCPRF at x, and of asking for updated public parameters for a random constraint g. Finally, $\mathcal{A}$ outputs a guess b'.

Game 1. In this game the challenger sets $\mathbf{a}_{i,1-u_i} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1,(i,1-u_i),\mathbf{0})$ for all $i \in [\ell]$, where $(u_1,\ldots,u_\ell) = (f_1^b \oplus d_1,\ldots,f_\ell^b \oplus d_\ell)$. All else is as in Game 0, including the ciphertexts $\mathbf{a}_{i,1-u_i}$.

Game 2. Game 2 is identical to Game 1 except for the ciphertexts of the form $\mathbf{b}_{j,1-\psi_j}$, which are computed as $\mathbf{b}_{j,1-\psi_j} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2,(j,1-\psi_j),\mathbf{0})$.

Claim 4.3.3.1. *If* ABE *is a selectively secure attribute-based encryption scheme, then Game 0 and Game 1 are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT adversary in the experiment of distinguishing Game 0 from Game 1. At the start of this experiment the challenger randomly chooses $\beta \leftarrow \{0,1\}$. The challenger and the adversary then play Game 0, if $\beta = 0$, or Game 1, if $\beta = 1$. Finally, $\mathcal{A}$ outputs a guess β' and wins if $\beta' = \beta$.

Given $\mathcal{A}$ we construct a PPT adversary $\mathcal{B}$ against ABE in the selective security game of Definition 4.1.1 as follows. $\mathcal{B}$ begins by running $\mathcal{A}$, which replies with two constraints $f^0, f^1 \in \{0,1\}^\ell$. Then it generates the public parameters $(\mathsf{BTVW.pp}, \{\mathbf{b}_{j,\beta}\}, \mathsf{ABE.pk}_2)$, the components $(\mathsf{BTVW.msk}, \{d_i\}, \mathsf{ABE.msk}_2)$ of the secret key and computes $\{\Psi_{i,\beta}\}$ as in the SCPRF key generation algorithm. It also chooses $b \leftarrow \{0,1\}$ uniformly.

The adversary $\mathcal{B}$ computes $(u_1,\ldots,u_\ell) = (f_1^b \oplus d_1,\ldots,f_\ell^b \oplus d_\ell)$ and sends to the ABE challenger the attribute sequence $\{(i,1-u_i)\}_{i\in[\ell]}$. Once the challenger has sampled $\beta \leftarrow \{0,1\}$ and generated $(\mathsf{ABE.msk}_1, \mathsf{ABE.pk}_1) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, $\mathcal{B}$ receives a key $\mathsf{ABE.pk}_1$ and then it sends the message pairs

$\{(\mu_{0,i}, \mu_{1,i})\}_{i\in[\ell]}$ given by $\mu_{0,i} = \Psi_{i,1-f_i^b}$ and $\mu_{1,i} = \mathbf{0}$. It then receives ciphertexts $\mathbf{a}_{i,1-u_i}$ which encrypt either $\Psi_{i,1-f_i^b}$ or $\mathbf{0}$, depending on the value of β.

Using the public key provided by the challenger, $\mathcal{B}$ computes $\mathbf{a}_{i,u_i} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1,(i,u_i),\Psi_{i,f_i^b})$ and it also generates $\mathsf{ABE.ck}_2$ as in the $\mathsf{SCPRF.Constrain}$ algorithm. Then it queries the ABE challenger on the constraint Φ_u. Since $\Phi_u(i,1-u_i)=1$ for all i, the query is allowed and the challenger must reply with a constrained key $\mathsf{ABE.ck}_1$. $\mathcal{B}$ sends updated public parameters $\mathsf{pp}' = (\mathsf{BTVW.pp}, \{\mathbf{a}_{i,\beta}\}, \{\mathbf{b}_{j,\beta}\}, \mathsf{ABE.pk}_1, \mathsf{ABE.pk}_2, u)$ and a constrained key $\mathsf{ck} = (\mathsf{ABE.ck}_1, \mathsf{ABE.ck}_2)$ to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a valid query x, $\mathcal{B}$ uses $\mathbf{s}$ to compute $y = \mathsf{BTVW.Eval}_{\mathsf{BTVW.pp}}(\mathsf{BTVW.msk}, x)$, which it sends as its reply. If $\mathcal{A}$ requests updated parameters, $\mathcal{B}$ samples $g \leftarrow \{0,1\}^\ell$, computes $u' = (g_1 \oplus d_1, \ldots, g_\ell \oplus d_\ell)$ and replies with (pp', u'). Once $\mathcal{A}$ outputs its guess β', $\mathcal{B}$ also outputs β'.

The game that $\mathcal{A}$ plays has the distribution of Game 0 if $\beta = 0$ and the distribution of Game 1 if $\beta = 1$. It follows that $\mathcal{B}$ wins against the ABE challenger with the same probability that $\mathcal{A}$ successfully distinguishes the two games. By selective security of ABE, $\mathcal{B}$ has negligible advantage, hence $\mathcal{A}$ has negligible advantage as well.

■

Claim 4.3.3.2. *If* ABE *is a selectively secure attribute-based encryption scheme, then Game 1 and Game 2 are computationally indistinguishable.*

The proof of Claim 4.3.3.2 is analogous to the proof of Claim 4.3.3.1, with the adversary $\mathcal{B}$ choosing the attributes (j, ψ_j) and the messages $\mu_{0,j} = \mathbf{c}_{j,1-\psi_j}$, $\mu_{1,j} = \mathbf{0}$. It follows that Games 0 and 2 are also indistinguishable under our security assumption for ABE.

Claim 4.3.3.3. *If* BTVW *is constraint-hiding, then any PPT adversary $\mathcal{A}$ has only negligible advantage in Game 2.*

Proof. Given an adversary $\mathcal{A}$ in Game 2, we build an adversary $\mathcal{B}$ against BTVW in the constraint hiding game of Definition 2.2.2 as follows. As its first step, $\mathcal{B}$ runs $\mathcal{A}$ in order to obtain constraints $f^0, f^1 \in \{0,1\}^\ell$, which it forwards to the challenger. At this point the challenger secretly flips a coin $b \leftarrow \{0,1\}$ and sends public parameters BTVW.pp and a constrained key $(\Psi, \{\mathbf{c}_j\})$ corresponding to f^b. $\mathcal{B}$ samples $u \leftarrow \{0,1\}^\ell$ and generates $(\mathsf{ABE.msk}_1, \mathsf{ABE.pk}_1) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, $(\mathsf{ABE.msk}_2, \mathsf{ABE.pk}_2) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$. Then it parses $\Psi = [\Psi_1 \mid \ldots \mid \Psi_\ell]$, calculates its binary decomposition $\psi_1, \ldots, \psi_L$ and computes

$$\begin{aligned}
\mathbf{a}_{i,u_i} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1,(i,u_i),\Psi_i),\\
\mathbf{a}_{i,1-u_i} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_1,(i,1-u_i),\mathbf{0}),\\
\mathbf{b}_{j,\psi_j} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2,(j,\psi_j),\mathbf{c}_j),\\
\mathbf{b}_{j,1-\psi_j} &\leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.pk}_2,(j,1-\psi_j),\mathbf{0}).
\end{aligned}$$

Using u and Ψ, $\mathcal{B}$ generates ABE constrained keys $\mathsf{ABE.ck}_1 \leftarrow \mathsf{ABE.Constrain}(\mathsf{ABE.msk}_1, \Phi_u)$ and $\mathsf{ABE.ck}_2 \leftarrow \mathsf{ABE.Constrain}(\mathsf{ABE.msk}_2, \Lambda_\Psi)$ and it sends the SCPRF public parameters $\mathsf{pp}' = (\mathsf{BTVW.pp}, \{\mathbf{a}_{i,\beta}\}, \{\mathbf{b}_{j,\beta}\}, \mathsf{ABE.pk}_1, \mathsf{ABE.pk}_2, u)$ and constrained key $(\mathsf{ABE.ck}_1, \mathsf{ABE.ck}_2)$ to $\mathcal{A}$.

For each query x that $\mathcal{A}$ makes, $\mathcal{B}$ forwards the query to the BTVW challenger and then forwards the replied value y back to $\mathcal{A}$. Whenever $\mathcal{A}$ requests an update of the parameters, $\mathcal{B}$ samples $u' \leftarrow \{0,1\}^\ell$ and responds with (pp', u'). Once $\mathcal{A}$ outputs a guess b', $\mathcal{B}$ halts and outputs b' as well.

Observe that in the game that $\mathcal{A}$ plays, as challenged by $\mathcal{B}$, the vector u is uniformly sampled from $\{0,1\}^\ell$, while in Game 2 it is computed as $u = (f_1^b \oplus d_1, \ldots, f_\ell^b \oplus d_\ell)$, where $d_1, \ldots, d_\ell \leftarrow \{0,1\}$. However, since $\mathcal{A}$ does not have access to $d_1, \ldots, d_\ell$ or functions of it other than u, in both cases u is uniformly distributed and independent from all else in the view of $\mathcal{A}$. Analogously, in both games all the vectors u' generated as parameter updates are uniform and independent from everything in the view of $\mathcal{A}$. Therefore the game that $\mathcal{A}$ plays has the distribution of Game 2.

By the security of BTVW, $\mathcal{B}$ has negligible advantage in the constraint-hiding game. Since $\mathcal{B}$ wins with the same probability with which $\mathcal{A}$ wins Game 2, $\mathcal{A}$ has negligible advantage in this game.

■

Finally, since Games 0 and 2 are computationally indistinguishable, from the last claim we conclude that the advantage of any adversary in Game 0 is negligible and that SCPRF is constraint-hiding.

□

4.4 Size of constrained keys

We proceed to show that our scheme SCPRF has succinct keys when instantiated with the ABE scheme of Boneh et al. [BGG$^+$14], in which the key size depends only on the depth of the circuit that computes a constraint.

Lemma 4.4.1. *The constraints Φ_u and Λ_Ψ featured in the $\mathrm{SCPRF.Constrain}$ algorithm are computable by circuits of depth $\mathcal{O}(\log \ell)$ and $\mathcal{O}(\log(\ell \operatorname{poly}(\lambda)))$, respectively, where ℓ is the description length of the constraint.*

Proof. The circuit for Φ_u receives as input (i, β), where $\beta \in \{0,1\}$ and $i \in [\ell]$ is represented in binary as $i_k \ldots i_1$, with $k = \lceil \log \ell \rceil$. With u hardcoded into it, the circuit computes u_i through a binary tree with k levels, as follows. On the first level it considers all the bits $u_1, \ldots, u_\ell$ (each corresponds to a node of the tree) and selects and keeps one from each pair (u_{2j-1}, u_{2j}) according to the input bit i_1, deleting the other. If $i_1 = 0$ (i is even) then u_{2j} is preserved, otherwise u_{2j-1} is kept (the j-th bit that survives can be computed as $(u_{2j-1} \wedge i_1) \vee (u_{2j} \wedge \neg i_1)$ for each j). On the second level, it selects one from each pair of remaining bits according to i_2, and so on until it is left with only one bit, which is u_i. Finally, the circuit outputs $u_i \oplus \beta$, which is equal to $\Phi_u(i, \beta)$.

The depth of this circuit is $\mathcal{O}(k) + 1 = \mathcal{O}(\log \ell)$. Analogously, there is a circuit that computes Λ_Ψ with depth $\mathcal{O}(\log |\Psi|) = \mathcal{O}(\log(\ell n^2 \log^2 q))$, where n, q are parameters of the BTVW private CPRF satisfying $n, \log q = \operatorname{poly}(\lambda)$.

□

From Theorem 4.1.2 and Lemma 4.4.1 we conclude that, when we use the ABE scheme of [BGG$^+$14] in our construction, we obtain a private CPRF with keys of size $\operatorname{poly}(\lambda, \log \ell)$. It follows that there exists a

polynomial $p(\cdot)$ such that, for all polynomial values of ℓ, the size of constrained keys is bounded by $p(\lambda)$ for sufficiently large λ. From this observation and the results of the previous section we obtain the next theorem, which is the main result of this chapter.

Theorem 4.4.2. *Consider the* SCPRF *construction from Section 4.2 with the attribute-based encryption scheme of [BGG+ 14] in place of* ABE*. Under the LWE assumption,* SCPRF *is a weak single-key private CPRF with updatable parameters in which the constrained keys are succinct.*

4.5 Separation between weak and strong security

Having proven that the constrained PRF presented in Section 4.2 has weak single-key selective security, we now show that it does not have strong single-key selective security. This implies that, although strong security implies weak security, the two notions are not equivalent.

In order to prove that this constrained PRF does not satisfy Definition 2.3.2, we present an attack in which the adversary uses the ability to obtain updated parameters for a constraint of its choice to obtain information about the circuit corresponding to its constrained key, thus breaking the constraint-hiding property. Intuitively, the scheme fails to achieve the stronger variant of constraint-hiding because the vector u which is added to the public parameters is a one-time pad encryption of the constraint f. When the adversary in the security game asks for the updated parameters for a constraint g of its choice, this is essentially a chosen plaintext attack, which breaks the one-time pad.

Theorem 4.5.1. *Let* SCPRF *be the constrained pseudorandom function described in Section 4.2. There exists a PPT adversary* $\mathcal{A}$ *that wins the constraint-hiding security game of Definition 2.3.2 against* SCPRF *with probability* 1*.*

Proof. We define $\mathcal{A}$ as follows. It selects any two constraints f^0 and f^1 that differ in the first bit, that is, $f^0_1 \neq f^1_1$ (for instance, $f^0 = (0, \ldots, 0)$, $f^1 = (1, \ldots, 1)$). In the query phase, after receiving public parameters $\mathsf{pp}' = (\mathsf{pp}, u)$ and a constrained key ck corresponding to f^b, $\mathcal{A}$ requests an update of the parameters for the constraint $g = f^0$. Once the challenger replies with parameters (pp', u'), the adversary $\mathcal{A}$ verifies if $u'_1 = u_1$ and, in that case, outputs $b' = 0$ as its guess. Otherwise, $\mathcal{A}$ outputs $b' = 1$.

In this interaction, the vectors u and u' are computed by the challenger as $u = (f^b_1 \oplus d_1, \ldots, f^b_\ell \oplus d_\ell)$ and $u' = (f^0_1 \oplus d_1, \ldots, f^0_\ell \oplus d_\ell)$. Therefore $u_1 = u'_1$ if and only if $f^b_1 = f^0_1$. Since $f^0_1 \neq f^1_1$, this occurs if and only if $b = 0$. It follows that, with total probability, $b' = b$ and $\mathcal{A}$ wins the game.

□

It is worth noting that Theorem 4.5.1 continues to hold even if we change the definition of strong single-key selective security to only allow queries of constraints g different from f^0 and f^1. In this case, the attacker $\mathcal{A}$ in the proof of the theorem should select g as any constraint such that $g_1 = f^0_1$ and $g \neq f^0$.

Chapter 5

Succinct-key private constrained PRF from functional encodings

In this chapter we propose an alternative private constrained pseudorandom function with updatable parameters with the property of succinct constrained keys. In this construction we use functional encodings and symmetric-key encryption to transform any private constrained PRF with single-key selective security into one that has succinct keys.

5.1 Symmetric-key encryption

Symmetric-key encryption (SKE) is a basic cryptographic primitive which allows the encryption and decryption of messages using the same key. A symmetric-key encryption scheme consists of PPT algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- $\mathsf{Setup}(1^\lambda)$ generates a key k.
- $\mathsf{Enc}(k, x)$ receives as input a key k and a string x and outputs a ciphertext c.
- $\mathsf{Dec}(k, c)$ receives as input a key k and a ciphertext c and outputs a string x.

Correctness. For any string x and any key $k \leftarrow \mathsf{Setup}(1^\lambda)$, we must have $\mathsf{Dec}(k, \mathsf{Enc}(k, x)) = x$.

The security definition that we will consider for SKE schemes is indistinguishability against chosen plaintext attacks (IND-CPA), which we present next. This is a simple and relatively weak notion that provides protection only against passive attackers, but it is sufficient for our purposes.

Definition 5.1.1 (IND-CPA)**.** *A symmetric-key encryption scheme* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be* IND-CPA secure *if in the following security game any stateful PPT adversary* $\mathcal{A}$ *has only negligible advantage, that is,* $\left|\Pr[b' = b] - \frac{1}{2}\right| = \mathrm{negl}(\lambda)$.

1. *The challenger generates* $k \leftarrow \mathsf{Setup}(1^\lambda)$ *and flips a coin* $b \leftarrow \{0, 1\}$.

2. *$\mathcal{A}$ can ask for encryptions of strings x, to which the challenger replies with $\mathsf{Enc}(k, x)$. Then $\mathcal{A}$ sends a pair of strings (x_0, x_1) to the challenger.*

3. *The challenger sends $c \leftarrow \mathsf{Enc}(k, x_b)$ to $\mathcal{A}$.*

4. *The adversary $\mathcal{A}$ may make additional encryption queries, as in step 2, and finally it outputs a guess $b' \in \{0, 1\}$.*

There are many possibilities for a concrete LWE-based SKE scheme to use in our CPRF construction. We consider the Regev cryptosystem [Reg05], which is a public-key encryption scheme, but may of course be seen as a symmetric-key system where the key consists of the both the public and the secret key.

Theorem 5.1.2 ([Reg05])**.** *Under the LWE assumption, there exists an IND-CPA secure SKE scheme, in which a plaintext of size k can be encrypted bit by bit to form a ciphertext of size $k \cdot \mathrm{poly}(\lambda)$.*

5.2 Functional encodings

Functional encodings (FE) were introduced by Wee and Wichs [WW21] as a tool for a candidate indistinguishability obfuscation construction. They allow a user to generate an encoding of a string x and a short "opening" for a function f, which can be decoded together to recover $f(x)$. A functional encoding scheme is a tuple $(\mathsf{Enc}, \mathsf{Open}, \mathsf{Dec})$ of PPT algorithms with the following syntax:

- $\mathsf{Enc}(1^\lambda, x)$ receives as input the security parameter λ and a string x and outputs an encoding c and the randomness r used in the process.
- $\mathsf{Open}(f, x, r)$ receives as input a function f, a string x and randomness r. It outputs an opening d.
- $\mathsf{Dec}(f, c, d)$ receives as input a function f, an encoding c and an opening d. It outputs a value y.

Correctness. For any string x, function f, and $(c, r) \leftarrow \mathsf{Enc}(1^\lambda, x)$, we must have

$$\mathsf{Dec}(f, c, \mathsf{Open}(f, x, r)) = f(x).$$

For our construction we require an FE scheme with one-opening simulation-based (1-SIM) security, as defined below. It is shown in [WW21] that such a scheme exists under the LWE assumption.

Definition 5.2.1 (1-SIM security)**.** *A functional encoding scheme $(\mathsf{Enc}, \mathsf{Open}, \mathsf{Dec})$ is said to be* 1-SIM secure *if there exists a PPT simulator Sim such that, for any PPT adversary $\mathcal{A}$, the distributions $\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}}$ and $\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathsf{Sim}}(1^\lambda)\}_{\lambda\in\mathbb{N}}$ described next are computationally indistinguishable.*

- $\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda) = (x, f, c, d)$*, where* $(x, f) \leftarrow \mathcal{A}(1^\lambda)$*,* $(c, r) \leftarrow \mathsf{Enc}(1^\lambda, x)$*,* $d \leftarrow \mathsf{Open}(f, x, r)$*.*
- $\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathsf{Sim}}(1^\lambda) = (x, f, c, d)$*, where* $(x, f) \leftarrow \mathcal{A}(1^\lambda)$*,* $(c, d) \leftarrow \mathsf{Sim}(1^\lambda, f, f(x))$*.*

Theorem 5.2.2 ([WW21])**.** *Under the LWE assumption, there exists a 1-SIM secure FE scheme in which the opening for a depth-t circuit $f : \{0, 1\}^z \to \{0, 1\}^k$ has size $\mathrm{poly}(\lambda, t, \log k)$, where λ is the security parameter.*

5.3 Succinct-key private constrained PRF

The core idea behind this construction is to take a private CPRF that does not have succinct keys, publish an encryption by a functional encoding scheme of its master secret key, and then give as the new constrained key an opening for the circuit that computes the constrained key for the original CPRF. In order to preserve the property of privacy, the constraint must be encrypted, and this encryption must be added to the public parameters so that the user can apply the FE decryption algorithm.

Let $\mathsf{CPRF} = (\mathsf{CPRF.KeyGen}, \mathsf{CPRF.Eval}, \mathsf{CPRF.Constrain}, \mathsf{CPRF.ConstrainEval})$ be a private constrained PRF. We assume that the probabilistic algorithm $\mathsf{CPRF.Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$ can be split into two procedures: a probabilistic algorithm $\mathsf{CPRF.Sample}_{\mathsf{pp}}(\mathsf{msk})$, which generates all the randomness R necessary to generate a constrained key, independently of the constraint f, and a deterministic algorithm $\mathsf{CPRF.Comp}_{\mathsf{pp}}(\mathsf{msk}, f, R)$, which computes a constrained key corresponding to f using the randomness R. We require that generating a key with these algorithms is equivalent to generating it with the usual constraining algorithm, that is, the outputs $\mathsf{ck} \leftarrow \mathsf{CPRF.Comp}_{\mathsf{pp}}(\mathsf{msk}, f, \mathsf{CPRF.Sample}_{\mathsf{pp}}(\mathsf{msk}))$ and $\mathsf{ck} \leftarrow \mathsf{CPRF.Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$ are equally distributed.

Let $\mathsf{FE} = (\mathsf{FE.Enc}, \mathsf{FE.Open}, \mathsf{FE.Dec})$ be a functional encoding scheme and let $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a symmetric-key encryption system. The following algorithms constitute our succinct-key private constrained PRF with updatable parameters SCPRF, which supports the same class of constraints as the underlying scheme CPRF.

- $\mathsf{SCPRF.KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$: The input parameters are the security parameter λ, the maximum description length ℓ of constraint functions, their input length z and their maximum depth t. To set up the scheme, generate $(\mathsf{CPRF.msk}, \mathsf{CPRF.pp}) \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$, $k \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and $R \leftarrow \mathsf{CPRF.Sample}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk})$. Then compute $(\mathbf{C}, r) \leftarrow \mathsf{FE.Enc}(1^\lambda, (\mathsf{CPRF.msk}, k, R))$. Output

  $$\mathsf{msk} = (\mathsf{CPRF.msk}, k, R, r)\,, \quad \mathsf{pp} = (\mathsf{CPRF.pp}, \mathbf{C})$$

  as the master secret key and the public parameters, respectively.

- $\mathsf{SCPRF.Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$: Output $y = \mathsf{CPRF.Eval}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, x)$.

- $\mathsf{SCPRF.Constrain}_{\mathsf{pp}}(\mathsf{msk}, f)$: Let $h \leftarrow \mathsf{SKE.Enc}(k, f)$. Consider the following circuit $\mathcal{C}_h$: on input a tuple (s, k, R), it computes $f = \mathsf{SKE.Dec}(k, h)$ and then outputs $\mathsf{CPRF.Comp}_{\mathsf{CPRF.pp}}(s, f, R)$. Compute $\mathbf{d} \leftarrow \mathsf{FE.Open}(\mathcal{C}_h, (\mathsf{CPRF.msk}, k, R), r)$ and output it as the constrained key for f. Output also $\mathsf{pp}' = (\mathsf{pp}, h)$ as the new public parameters.

- $\mathsf{SCPRF.ConstrainEval}_{\mathsf{pp}'}(\mathbf{d}, x)$: Retrieve h from the parameters and compute $\mathsf{ck} \leftarrow \mathsf{FE.Dec}(\mathcal{C}_h, \mathbf{C}, \mathbf{d})$. Output $y = \mathsf{CPRF.ConstrainEval}_{\mathsf{CPRF.pp}}(\mathsf{ck}, x)$.

5.4 Proof of security

In this section we prove that the private constrained PRF with updatable parameters SCPRF described above has strong single-key selective security, as per Definition 2.3.2.

Theorem 5.4.1 (Correctness)**.** *Let* CPRF *be a correct constrained PRF,* FE *a correct functional encoding scheme and* SKE *a correct symmetric-key encryption system. Then* SCPRF *is also correct.*

Proof. Since Eval depends only on the parameters CPRF.pp, we have $\mathsf{Eval}_{\mathsf{pp}'}(\mathsf{msk}, x) = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$ for any $x \in \{0,1\}^z$. Let f be a constraint. By the correctness of the symmetric-key encryption scheme, $\mathsf{SKE.Dec}(k, h) = f$, hence $\mathcal{C}_h(\mathsf{CPRF.msk}, k, R) = \mathsf{CPRF.Comp}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f, R)$. Therefore, by the correctness of the functional encoding scheme, the constrained key ck computed during the algorithm SCPRF.ConstrainEval is a properly generated constrained key of the CPRF scheme for the predicate f. It follows that $\mathsf{SCPRF.ConstrainEval}_{\mathsf{pp}'}(\mathbf{d}, x) = \mathsf{SCPRF.Eval}_{\mathsf{pp}}(\mathsf{msk}, x)$ holds if and only if $\mathsf{CPRF.ConstrainEval}_{\mathsf{CPRF.pp}}(\mathsf{ck}, x) = \mathsf{CPRF.Eval}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, x)$, and the latter occurs whenever $f(x) = 0$ since CPRF is correct. Therefore SCPRF is also correct.

□

Theorem 5.4.2 (Pseudorandomness)**.** *Let* CPRF *be a constrained PRF satisfying pseudorandomness at constrained points (as per Definition 2.2.2),* FE *a 1-SIM secure functional encoding scheme and* SKE *a correct symmetric-key encryption system. Then* SCPRF *satisfies the pseudorandomness at constrained points property of Definition 2.3.2.*

Proof. Consider the following two security games, where Sim is a PPT simulator with the properties described in Definition 5.2.1, which exists by the assumption that the functional encoding scheme FE is 1-SIM secure.

Game 0. This is the pseudorandomness security game of Definition 2.3.2 of the adversary $\mathcal{A}$ against SCPRF.

Game 1. In this game, instead of computing the encoding $\mathbf{C}$ and the constrained key $\mathbf{d}$ as usual, the challenger sets $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_h, \mathsf{ck})$, where $\mathsf{ck} \leftarrow \mathsf{CPRF.Comp}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f, R)$ and $h \leftarrow \mathsf{SKE.Enc}(k, f)$. Game 1 is otherwise identical to Game 0.

Observe that in Game 1 we have $\mathcal{C}_h(\mathsf{CPRF.msk}, k, R) = \mathsf{ck}$.

Claim 5.4.2.1. *If* FE *is a* 1-SIM *secure functional encoding scheme, then Game 0 and Game 1 are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT adversary in the experiment of distinguishing Game 0 from Game 1. At the start of this experiment the challenger randomly chooses $\beta \leftarrow \{0,1\}$. The challenger and the adversary then play Game 0, if $\beta = 0$, or Game 1, if $\beta = 1$. Finally, $\mathcal{A}$ outputs a guess β' and wins if $\beta' = \beta$.

Given $\mathcal{A}$ we construct a PPT adversary $\mathcal{B}$ that distinguishes the two experiments $\mathsf{Exp}^{\mathsf{real}}_{\mathcal{B}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{B},\mathsf{Sim}}(1^\lambda)$ of Definition 5.2.1 as follows. $\mathcal{B}$ begins by running $\mathcal{A}$, which outputs a constraint f. Then it generates $(\mathsf{CPRF.msk}, \mathsf{CPRF.pp}) \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$, $R \leftarrow \mathsf{CPRF.Sample}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk})$ and a key

$k \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. $\mathcal{B}$ computes $h \leftarrow \mathsf{SKE.Enc}(k, f)$ and sends $((\mathsf{CPRF.msk}, k, R), \mathcal{C}_h)$ to the challenger as the FE input and function pair.

At this point the challenger samples $\beta \leftarrow \{0,1\}$, which determines if it will act according to the real or simulated experiment. It sends to $\mathcal{B}$ a pair $(\mathbf{C}, \mathbf{d})$, where $(\mathbf{C}, r) \leftarrow \mathsf{FE.Enc}(1^\lambda, (\mathsf{CPRF.msk}, k, R))$ and $\mathbf{d} \leftarrow \mathsf{FE.Open}(\mathcal{C}_h, (\mathsf{CPRF.msk}, k, R), r)$, if $\beta = 0$, or $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_h, \mathsf{ck})$, if $\beta = 1$. Here $\mathsf{ck} = \mathcal{C}_h(\mathsf{CPRF.msk}, k, R)$. $\mathcal{B}$ flips a coin $b \leftarrow \{0,1\}$ and sends SCPRF public parameters $\mathsf{pp}' = (\mathsf{CPRF.pp}, \mathbf{C}, h)$ and constrained key $\mathbf{d}$ to $\mathcal{A}$.

For each valid query x that $\mathcal{A}$ makes, $\mathcal{B}$ responds with $y = \mathsf{CPRF.Eval}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, x)$, if $b = 0$, or with $y \leftarrow \mathbb{Z}_p$, if $b = 1$. Whenever $\mathcal{A}$ queries a constraint g, $\mathcal{B}$ computes $h' \leftarrow \mathsf{SKE.Enc}(k, g)$ and sends to $\mathcal{A}$ the updated parameters (pp', h'). Finally, $\mathcal{A}$ outputs a guess b' for b and a guess β' for whether it played Game 0 or Game 1. The output of $\mathcal{B}$ is β'.

Observe that when $\beta = 0$ the game that $\mathcal{A}$ is playing is distributed exactly like Game 0 and when $\beta = 1$ it is distributed like Game 1. Therefore $\mathcal{B}$ has the same advantage in distinguishing $\mathsf{Exp}^{\mathsf{real}}_{\mathcal{B}}(1^\lambda)$ from $\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{B},\mathsf{Sim}}(1^\lambda)$ that $\mathcal{A}$ has in distinguishing Game 0 from Game 1. By 1-SIM security of FE, $\mathcal{B}$ has negligible advantage, hence $\mathcal{A}$ also has negligible advantage.

■

Claim 5.4.2.2. *If* CPRF *is pseudorandom, then any adversary in Game 1 has only negligible advantage.*

Proof. Let $\mathcal{A}$ be an adversary against SCPRF in Game 1. We construct as follows an adversary $\mathcal{B}$ against CPRF in the pseudorandomness game of Definition 2.2.2. The game begins with $\mathcal{B}$ running $\mathcal{A}$ to receive a constraint $f \in \{0,1\}^\ell$, which it forwards to the CPRF challenger. After receiving public parameters CPRF.pp and a constrained key ck from the challenger, $\mathcal{B}$ generates $k \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and computes $h \leftarrow \mathsf{SKE.Enc}(k, f)$ and $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_h, \mathsf{ck})$. Then $\mathcal{B}$ sends to $\mathcal{A}$ SCPRF public parameters $\mathsf{pp}' = (\mathsf{CPRF.pp}, \mathbf{C}, h)$ and a constrained key $\mathbf{d}$.

At this point $\mathcal{B}$ must be able to reply to the queries of $\mathcal{A}$. Whenever $\mathcal{A}$ queries a point $x \in \{0,1\}^z$ such that $f(x) = 1$, $\mathcal{B}$ makes that same query to the challenger. The challenger responds with a value y, which $\mathcal{B}$ forwards to $\mathcal{A}$ as its reply. If $\mathcal{A}$ asks for updated parameters for a constraint g, $\mathcal{B}$ computes $h' \leftarrow \mathsf{SKE.Enc}(k, g)$ and replies with (pp', h'). Once $\mathcal{A}$ halts and outputs a guess b', $\mathcal{B}$ outputs b' as well.

Observe that $\mathcal{B}$ wins the game against CPRF if and only if $\mathcal{A}$ wins Game 1 against SCPRF. By pseudorandomness of the CPRF protocol, $\mathcal{B}$ has negligible advantage, hence $\mathcal{A}$ also has negligible advantage in Game 1. ■

Since Game 0 and Game 1 are computationally indistinguishable, it follows that the advantage of $\mathcal{A}$ in Game 0 is negligible, that is, $|\Pr[b' = b] - \frac{1}{2}| = \mathsf{negl}(\lambda)$ in Game 0.

□

Theorem 5.4.3 (Constraint hiding)**.** *Let* CPRF *be a constrained PRF with the constraint hiding property (as per Definition 2.2.2),* FE *a 1-SIM secure functional encoding scheme and* SKE *a IND-CPA secure symmetric-key encryption system. Then* SCPRF *is constraint-hiding, as per Definition 2.3.2.*

Proof. The proof consists of the following sequence of hybrid security games, with the first and last games being the cases $b = 0$ and $b = 1$ of the SCPRF constraint hiding game, respectively. We show that each game is computationally indistinguishable from the previous one.

Game 0. This is the constraint hiding security game of Definition 2.3.2 for the SCPRF protocol, in the case that the challenger chooses $b = 0$. The game begins with the adversary $\mathcal{A}$ submitting two constraints f^0, f^1. Then the challenger generates $(\mathsf{CPRF.msk}, \mathsf{CPRF.pp}) \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda, 1^\ell, 1^z, 1^t)$ and sends public parameters $\mathsf{pp}' = (\mathsf{CPRF.pp}, \mathbf{C}, h_0)$ and a constrained key $\mathbf{d}$ to $\mathcal{A}$, where $(\mathbf{C}, r) \leftarrow \mathsf{FE.Enc}(1^\lambda, (\mathsf{CPRF.msk}, k, R))$, $h_0 \leftarrow \mathsf{SKE.Enc}(k, f^0)$, $\mathbf{d} \leftarrow \mathsf{FE.Open}(\mathcal{C}_{h_0}, (\mathsf{CPRF.msk}, k, R), r)$ and $R \leftarrow \mathsf{CPRF.Sample}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk})$. $\mathcal{A}$ performs additional computations while being able to query points x such that $f^0(x) = f^1(x)$, to the which challenger replies with $y = \mathsf{CPRF.Eval}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, x)$, and constraint queries g, to which the challenger responds with corresponding updated parameters. We omit the guessing step at the end of the game.

Game 1. Instead of defining $\mathbf{C}$ and $\mathbf{d}$ as in Game 0, in this game the challenger computes them as $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_{h_0}, \mathsf{ck}_0)$, where Sim is the simulator from Definition 5.2.1 for the FE scheme and $\mathsf{ck}_0 \leftarrow \mathsf{CPRF.Constrain}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f^0, R)$.

Game 2. In this game the challenger sends $\mathsf{pp}' = (\mathsf{CPRF.pp}, \mathbf{C}, h_1)$ as public parameters and $\mathbf{d}$ as the constrained key , where $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_{h_1}, \mathsf{ck}_0)$, $\mathsf{ck}_0 \leftarrow \mathsf{CPRF.Constrain}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f^0, R)$ and $h_1 \leftarrow \mathsf{SKE.Enc}(k, f^1)$.

Game 3. In this game the challenger sends $\mathsf{pp}' = (\mathsf{CPRF.pp}, \mathbf{C}, h_1)$ as public parameters and $\mathbf{d}$ as the constrained key , where $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_{h_1}, \mathsf{ck}_1)$, $\mathsf{ck}_1 \leftarrow \mathsf{CPRF.Constrain}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f^1, R)$ and $h_1 \leftarrow \mathsf{SKE.Enc}(k, f^1)$.

Game 4. This is the constraint hiding security game of Definition 2.2.2 for the SCPRF protocol, in the case that the challenger chooses $b = 1$. The difference between Games 3 and 4 is that in Game 3 the variables $\mathbf{C}$ and $\mathbf{d}$ are generated as $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_{h_1}, \mathsf{ck}_1)$, while in Game 4 they are given by $(\mathbf{C}, r) \leftarrow \mathsf{FE.Enc}(1^\lambda, (\mathsf{CPRF.msk}, k, R))$ and $\mathbf{d} \leftarrow \mathsf{FE.Open}(\mathcal{C}_{h_1}, (\mathsf{CPRF.msk}, k, R), r)$. Here $h_1 \leftarrow \mathsf{SKE.Enc}(k, f^1)$ and $\mathsf{ck}_1 \leftarrow \mathsf{CPRF.Constrain}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f^1, R)$.

Claim 5.4.3.1. *If* FE *is a* 1-SIM *secure functional encoding scheme, then Game 0 and Game 1 are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT adversary in the experiment of distinguishing Game 0 from Game 1. At the start of this experiment the challenger randomly chooses $\beta \leftarrow \{0, 1\}$. The challenger and the adversary then play Game 0, if $\beta = 0$, or Game 1, if $\beta = 1$. Finally, $\mathcal{A}$ outputs a guess β' and wins if $\beta' = \beta$.

Given $\mathcal{A}$ we construct a PPT adversary $\mathcal{B}$ that distinguishes the experiments $\mathsf{Exp}^{\mathsf{real}}_{\mathcal{B}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{B},\mathsf{Sim}}(1^\lambda)$ of Definition 5.2.1 as follows. $\mathcal{B}$ begins by simulating $\mathcal{A}$ to obtain a pair of constraints f^0, f^1. Then it generates $(\mathsf{CPRF.msk}, \mathsf{CPRF.pp}) \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$, $k \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and computes $h_0 \leftarrow \mathsf{SKE.Enc}(k, f^0)$. $\mathcal{B}$ samples $R \leftarrow \mathsf{CPRF.Sample}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk})$ and sends $((\mathsf{CPRF.msk}, k, R), \mathcal{C}_{h_0})$ to the challenger as the FE input and function pair.

At this point the challenger samples $\beta \leftarrow \{0,1\}$, which determines if it will act according to the real or simulated experiment. It sends to $\mathcal{B}$ a pair $(\mathbf{C}, \mathbf{d})$, where $(\mathbf{C}, r) \leftarrow \mathsf{FE.Enc}(1^\lambda, (\mathsf{CPRF.msk}, k, R))$ and $\mathbf{d} \leftarrow \mathsf{FE.Open}(\mathcal{C}_{h_0}, (\mathsf{CPRF.msk}, k, R), r)$, if $\beta = 0$, or $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_{h_0}, \mathsf{ck}_0)$, if $\beta = 1$. Here $\mathsf{ck}_0 = \mathcal{C}_{h_0}(\mathsf{CPRF.msk}, k, R)$. $\mathcal{B}$ sends SCPRF public parameters $(\mathsf{CPRF.pp}, \mathbf{C}, h_0)$ and constrained key $\mathbf{d}$ to $\mathcal{A}$.

For each valid query x that $\mathcal{A}$ makes, $\mathcal{B}$ responds with $y = \mathsf{CPRF.Eval}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, x)$. Whenever $\mathcal{A}$ queries a constraint g, $\mathcal{B}$ computes $h' \leftarrow \mathsf{SKE.Enc}(k, g)$ and sends to $\mathcal{A}$ the updated parameters (pp', h'). When $\mathcal{A}$ halts and outputs a guess β', $\mathcal{B}$ outputs β' as well.

Observe that when $\beta = 0$ the game that $\mathcal{A}$ is playing is distributed like Game 0 and when $\beta = 1$ it is distributed like Game 1. Therefore $\mathcal{B}$ has the same advantage in distinguishing $\mathsf{Exp}^{\mathsf{real}}_{\mathcal{B}}(1^\lambda)$ from $\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{B},\mathsf{Sim}}(1^\lambda)$ that $\mathcal{A}$ has in distinguishing Game 0 from Game 1. By 1-SIM security of FE, $\mathcal{B}$ has negligible advantage, hence $\mathcal{A}$ also has negligible advantage.

■

Claim 5.4.3.2. *If* SKE *is an IND-CPA secure symmetric-key encryption scheme, then Game 1 and Game 2 are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT adversary that distinguishes Game 1 from Game 2. We construct a PPT adversary $\mathcal{B}$ for the IND-CPA security game against SKE with the same advantage as $\mathcal{A}$.

$\mathcal{B}$ starts by running $\mathcal{A}$ to obtain two constraints, f^0 and f^1, which it sends to the IND-CPA challenger as the challenge plaintexts. After generating $k \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and choosing $\beta \leftarrow \{0,1\}$, the challenger sends $h \leftarrow \mathsf{SKE.Enc}(k, f^\beta)$ to $\mathcal{B}$. Then $\mathcal{B}$ generates $(\mathsf{CPRF.msk}, \mathsf{CPRF.pp}) \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$, $R \leftarrow \mathsf{CPRF.Sample}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk})$, and it computes $\mathsf{ck}_0 = \mathsf{CPRF.Constrain}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f^0, R)$, $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_h, \mathsf{ck}_0)$. It sends public parameters $(\mathsf{CPRF.pp}, \mathbf{C}, h)$ and a constrained key $\mathbf{d}$ to $\mathcal{A}$.

In the query phase, $\mathcal{B}$ replies with $y = \mathsf{CPRF.Eval}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, x)$ for each string x submitted by $\mathcal{A}$. Whenever $\mathcal{A}$ queries a constraint g, $\mathcal{B}$ forwards g to the SKE challenger as an encryption query and receives a ciphertext h'. Then it sends the updated parameters (pp', h') to the adversary $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess β', with $\beta' = 0$ and $\beta' = 1$ corresponding to Game 1 and Game 2, respectively. The output of $\mathcal{B}$ is also β'.

The game that $\mathcal{A}$ plays against $\mathcal{B}$ is distributed like Game 1 if $\beta = 0$ and like Game 2 if $\beta = 1$. It follows that $\mathcal{A}$ and $\mathcal{B}$ have the same advantage in their respective games. By IND-CPA security of SKE, the advantage of $\mathcal{B}$ is a negligible function of λ, hence so is the advantage of $\mathcal{A}$.

■

Claim 5.4.3.3. *If* CPRF *is a constrained PRF satisfying the constraint hiding property, then Game 2 and Game 3 are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT adversary that distinguishes Game 2 from Game 3. We show that there exists a PPT adversary $\mathcal{B}$ for the constraint hiding security game of Definition 2.2.2 against CPRF with the same advantage as $\mathcal{A}$.

The adversary $\mathcal{B}$ operates as follows. It begins by requesting constraints f^0, f^1 from $\mathcal{A}$, which it then forwards to the challenger. After generating $(\mathsf{CPRF.msk}, \mathsf{CPRF.pp}) \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$ and choosing $\beta \leftarrow \{0,1\}$, the challenger sends $\mathsf{ck} \leftarrow \mathsf{CPRF.Constrain}_{\mathsf{CPRF.pp}}(\mathsf{CPRF.msk}, f^\beta)$ to $\mathcal{B}$. At this point $\mathcal{B}$ generates $k \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and computes $h_1 \leftarrow \mathsf{SKE.Enc}(k, f^1)$, $(\mathbf{C}, \mathbf{d}) \leftarrow \mathsf{Sim}(1^\lambda, \mathcal{C}_{h_1}, \mathsf{ck})$. $\mathcal{B}$ then sends public parameters $(\mathsf{CPRF.pp}, \mathbf{C}, h_1)$ and constrained key $\mathbf{d}$ to $\mathcal{A}$.

Whenever $\mathcal{A}$ queries a point x, $\mathcal{B}$ forwards that query to the challenger and then forwards the replied value y back to $\mathcal{A}$. If $\mathcal{A}$ queries a constraint g, $\mathcal{B}$ computes $h' \leftarrow \mathsf{SKE.Enc}(k, g)$ and replies with the updated parameters (pp', h'). After the query phase, $\mathcal{A}$ outputs a guess β', with $\beta' = 0$ and $\beta' = 1$ corresponding to Game 2 and Game 3, respectively. $\mathcal{B}$ also outputs β'.

Observe that when $\beta = 0$ the game that $\mathcal{A}$ plays has the correct distribution of Game 2, and when $\beta = 1$ it has the distribution of Game 3. Consequently, the advantage of $\mathcal{B}$ in distinguishing a CPRF constrained key for f^0 from a constrained key for f^1 is equal to the advantage of $\mathcal{A}$ in distinguishing Game 2 from Game 3. By the security of CPRF, both $\mathcal{B}$ and $\mathcal{A}$ have negligible advantage.

■

Claim 5.4.3.4. *If* FE *is a* 1-SIM *secure functional encoding scheme, then Game 3 and Game 4 are computationally indistinguishable.*

The proof of Claim 5.4.3.4 is analogous to the proof of Claim 5.4.3.1, with f^1 in place of f^0. We have shown that the cases $b = 0$ and $b = 1$ in the constraint hiding game of Definition 2.2.2, corresponding to Game 0 and Game 4, respectively, are computationally indistinguishable. Therefore the SCPRF protocol is constraint-hiding.

□

5.5 Size of constrained keys

In this section we consider our private constrained PRF with updatable parameters SCPRF from Section 5.3 to be instantiated with the following schemes as building blocks:

- The BTVW private constrained PRF [BTVW17] in place of CPRF;
- The functional encoding scheme of Wee and Wichs [WW21] in place of FE;
- The Regev cryptosystem [Reg05] in place of SKE, with encryption and decryption performed bit by bit.

Observe that the algorithm $\mathrm{BTVW.Constrain}$ (see Section 3.4) can easily be split into a probabilistic part $\mathrm{BTVW.Sample}$ and a deterministic part $\mathrm{BTVW.Comp}$, as described next.

- $\mathrm{BTVW.Sample}_{\mathrm{pp}}(\mathrm{msk})$: Sample vectors $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_\ell \leftarrow \chi^{(n+1)\log q}$ and matrices $\mathbf{R}_1, \ldots, \mathbf{R}_\ell \leftarrow \{0,1\}^{(n+1)\log q \times (n+1)\log q}$. Output $R = (\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_\ell, \mathbf{R}_1, \ldots, \mathbf{R}_\ell)$.

- $\mathrm{BTVW.Comp}_{\mathrm{pp}}(\mathrm{msk}, f, R)$: Compute

$$\Psi_i = \begin{pmatrix} \mathbf{B} \\ \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}_0^\mathsf{T} \end{pmatrix} \mathbf{R}_i + f_i \mathbf{G}$$

for all $i \in [\ell]$ and let $(\psi_1, \ldots, \psi_L)$ be the binary representation of $\Psi = [\Psi_1 \mid \ldots \mid \Psi_\ell]$. Compute also $\mathbf{c}_j^\mathsf{T} = \mathbf{s}^\mathsf{T}(\mathbf{B}_j - \psi_j \overline{\mathbf{G}}) + \mathbf{e}_j^\mathsf{T}$ for all $j \in [L]$ and output $\mathsf{ck} = (\Psi, \{\mathbf{c}_j\}_{j \in [L]})$.

Lemma 5.5.1. *The depth of the circuit $\mathcal{C}_h$ featured in the $\mathrm{SCPRF.Constrain}$ algorithm is independent from the constraint description length.*

Proof. We denote by ℓ the size of the description of the constraint f. Recall that $\mathcal{C}_h$ receives as input (s, k, R) and its first step is to compute $f = \mathrm{SKE.Dec}(k, h)$. Since each of the ℓ bits of the description of f was encrypted separately to obtain h, the ℓ blocks of h can be decrypted in parallel. Since the decryption of each block is independent from ℓ, this step can be performed in depth $\mathrm{poly}(\lambda)$.

The second and final step is to compute $\mathrm{ck} = \mathrm{CPRF.Comp}(s, f, R)$. If we observe the description of the $\mathrm{BTVW.Comp}$ algorithm above, we see that the matrices $\Psi_1, \ldots, \Psi_\ell$ can be computed in parallel, resulting in an additional depth of $\mathrm{poly}(\lambda)$, independently from ℓ. We may assume that these matrices are represented column by column, in which case the binary representation of Ψ consists simply of the binary representations of $\Psi_1, \ldots, \Psi_\ell$. Finally, the $L = \ell\,\mathrm{poly}(\lambda)$ vectors $\mathbf{c}_1, \ldots, \mathbf{c}_L$ are individually independent from ℓ and can be computed in parallel, thus adding depth $\mathrm{poly}(\lambda)$ to our circuit. The total depth of $\mathcal{C}_h$ is therefore $\mathrm{poly}(\lambda)$ and independent from ℓ.

□

Lemma 5.5.2. *The size of the opening $\mathbf{d}$ in the $\mathrm{SCPRF.Constrain}$ algorithm is $\mathrm{poly}(\lambda, \log \ell)$, where ℓ is the constraint description length.*

Proof. By Theorem 5.2.2, the size of the opening is $\mathrm{poly}(\lambda, t, \log k)$, where t is the depth of the circuit $\mathcal{C}_h$ and k is the size of its output. By Lemma 5.5.1, $t = \mathrm{poly}(\lambda)$. On the other hand, since the output of $\mathcal{C}_h$ is a BTVW constrained key, we have $k = |(\Psi, \mathbf{c}_1, \ldots, \mathbf{c}_L)| = \mathcal{O}(\ell n^3 \log^4 q) = \mathcal{O}(\ell\,\mathrm{poly}(\lambda))$. We conclude that $|\mathbf{d}| = \mathrm{poly}(\lambda, \log \ell)$. □

Since the size of the constrained keys is $\mathrm{poly}(\lambda, \log \ell)$, we conclude that there exists a polynomial $p(\cdot)$ such that, for all polynomial values of ℓ, the key size is asymptotically bounded by $p(\lambda)$, independently of ℓ. We summarize the results of this chapter in the following theorem.

Theorem 5.5.3. *Consider the SCPRF construction from Section 5.3 instantiated with the private CPRF of [BTVW17], the FE scheme of [WW21] and the SKE scheme of [Reg05]. Under the LWE assumption, SCPRF is a strong single-key private CPRF with updatable parameters in which the constrained keys are succinct.*

Chapter 6

Conclusions

Modern cryptography encompasses a multitude of different protocols designed for varied purposes. In this thesis we study a very particular system with interesting applications, which is the private constrained PRF. We focus on the aspect of the size of constrained keys and on the question of having it be independent from the constraints. We partially solve this problem through the introduction of CPRFs with updatable parameters.

Our main contribution is a succinct-key private constrained PRF with updatable parameters. Of the two constructions we presented, the second appears to be superior due to its simplicity and stronger security, but the first may also hold some value in practice. Both are based on learning with errors, a standard cryptographic assumption which also provides quantum resistance.

An idea for possible future work is to attempt to obtain variants with succinct keys for other types of private CPRFs, namely with different security requirements (e.g. adaptive security, multi-key security) or for more restricted classes of constraints, or for similar protocols, such as private programmable PRFs [BLW17]. At first glance, our second proposed construction, featuring functional encodings, appears to have potential in this regard. Since it uses a generic private CPRF, it could perhaps be used to upgrade any private CPRF scheme into a scheme that has succinct keys and a similar level of security. Moreover, we have shown that it is secure when the updated parameters are adversarially chosen, so it could potentially even achieve this when considering multi-key security. However, some problems would certainly arise depending on the security definition. For instance, the technique used in our security proof fails if we consider adaptive instead of selective security, as the encoding C would have to be generated before knowing the constraint chosen by the adversary.

Another very interesting project would be to investigate the possibility of having private CPRFs with succinct keys without resorting to a generalized definition such as ours. The reason why we found the standard definition insufficient and considered a broader one is that, in the primitives we currently have which restrict access to encrypted data and have short keys (such as ABE and FE), in order to use a key for decryption one needs to know the circuit with which the key was generated, and this depends on the constraint. This is not a problem for non-private CPRFs, because in their case the constraint is assumed to be known by the holder of the constrained key but is not considered to be part of the key.

The user has therefore access to additional information that is not reflected in the size of the key. Our solution mimics this situation by placing the additional information in the public parameters. We are very curious as to whether or not achieving succinct keys with the usual definition is possible.

Bibliography

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, pages 533–556, 2014.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2), 2012.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.

[BKM17] Dan Boneh, Sam Kim, and Hart Montgomery. Private puncturable PRFs from standard lattice assumptions. In *EUROCRYPT*, pages 415–445, 2017.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *PKC*, 2017.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *TCC*, pages 264–302, 2017.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, pages 1–30, 2015.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, pages 280–300, 2013.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In *EUROCRYPT*, pages 446–476, 2017.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, , and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, pages 1115–1127, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.

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

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *CCS*, pages 669–684, 2013.

[KW19] Sam Kim and David J. Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In *CRYPTO*, pages 335–366, 2019.

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

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 333–342, 2009.

[Pei16] Chris Peikert. A decade of lattice cryptography. In *Foundations and Trends in Theoretical Computer Science*, 2016.

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *PKC*, pages 675–701, 2018.

[QWZ18] Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking PRFs under standard assumptions: Public marking and security with extraction queries. In *TCC*, pages 669–698, 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93, 2005.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *SIAM J. Comput.*, pages 1484–1509, 1997.

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

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In *EUROCRYPT*, pages 127–156, 2021.

[YAYX20] Rupeng Yang, Man Ho Au, Zuoxia Yu, and Qiuliang Xu. Collusion resistant watermarkable prfs from standard assumptions. In *CRYPTO*, pages 590–620, 2020.