Constrained Key-Homomorphic PRFs from Standard Lattice Assumptions
(Or: How to Secretly Embed a Circuit in Your PRF)*

Zvika Brakerski[1,**] and Vinod Vaikuntanthan[2,***]

[1] Weizmann Institute of Science, Rehovot, Israel
zvika.brakerski@weizmann.ac.il
[2] Massachusetts Institute of Technology, Cambridge, MA, USA
vinodv@csail.mit.edu

Abstract. Boneh et al. (Crypto 13) and Banerjee and Peikert (Crypto 14) constructed pseudorandom functions (PRFs) from the Learning with Errors (LWE) assumption by embedding combinatorial objects, a path and a tree respectively, in instances of the LWE problem. In this work, we show how to generalize this approach to embed *circuits*, inspired by recent progress in the study of Attribute Based Encryption.

Embedding a universal circuit for some class of functions allows us to produce *constrained keys* for functions in this class, which gives us the first standard-lattice-assumption-based constrained PRF (CPRF) for general bounded-description bounded-depth functions, for arbitrary polynomial bounds on the description size and the depth. (A constrained key w.r.t a circuit C enables one to evaluate the PRF on all x for which $C(x) = 1$, but reveals nothing on the PRF values at other points.) We rely on the LWE assumption and on the one-dimensional SIS (Short Integer Solution) assumption, which are both related to the worst case hardness of general lattice problems. Previous constructions for similar function classes relied on such exotic assumptions as the existence of multilinear maps or secure program obfuscation. The main drawback of our construction is that it does not allow collusion (i.e. to provide more than a single constrained key to an adversary). Similarly to the aforementioned previous works, our PRF family is also *key homomorphic*.

Interestingly, our constrained keys are very short. Their length does not depend directly either on the size of the constraint circuit or on the input length. We are not aware of any prior construction achieving this property, even relying on strong assumptions such as indistinguishability obfuscation.

* An extended version of this manuscript can be found in [11].

** Supported by the Israel Science Foundation (Grant No. 468/14) and by the Alon Young Faculty Fellowship.

*** Research supported by DARPA Grant number FA8750-11-2-0225, Alfred P. Sloan Research Fellowship, NSF CAREER Award CNS-1350619, NSF Frontier Grant CNS-1414119, Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

Y. Dodis and J.B. Nielsen (Eds.): TCC 2015, Part II, LNCS 9015, pp. 1–30, 2015.

1 Introduction

A pseudorandom function family (PRF) [14] is a finite set of functions $\{F_s : D \to R\}_s$, indexed by a seed (or key) s, such that for a random s, F_s is efficiently computable given s, and is computationally indistinguishable from a random function from D to R, given oracle access. Since the introduction of this concept, PRFs have been one of the most fundamental building blocks in cryptography. Many variants of PRFs with additional properties have been introduced and have found a plethora of applications in cryptography. In this work, we will focus on *Constrained PRFs* and *Key-Homomorphic PRFs*.

Constrained PRFs. Constrained PRFs (CPRFs) have been introduced simultaneously by Boneh and Waters [9], Kiayias et al. [18] (as "Delegatable PRFs") and by Boyle, Goldwasser and Ivan [10] (as "Functional PRFs"). Here an adversary is allowed to ask for a constrained key which should allow it to evaluate the PRF on a subset of the inputs, while revealing nothing about the values at other inputs. It has been shown [9,18,10] how to construct CPRFs for function classes of the form $x \in [i, j]$ (where the input is interpreted as an integer) based on any one-way function. This in particular allows for the "puncturing" technique of Sahai and Waters [26] that found many uses in the obfuscation literature. Further, [9] showed how to achieve more complicated function classes such as bit fixing functions and even arbitrary circuits, but those require use of cryptographic multilinear maps. They also introduce a number of applications for such CPRFs, including broadcast encryption schemes and identity based key exchange. Hofheinz et al. [17] show how to achieve adaptively secure CPRFs from indistinguishability obfuscation using a random oracle.

The original definition of CPRFs requires resilience to arbitrary collusion. Namely, a constrained key for C_1, C_2 should give no more information than a constrained key for $C_1 \vee C_2$ and must not reveal anything about values where $C_1(x) = C_2(x) = \mathsf{false}$. Many of the applications of CPRFs (e.g. for broadcast encryption and identity based key exchange) rely on collusion resilience. Unfortunately, our construction in this work will not allow collusions, and therefore will not be useful for these applications. We hope that future works will be able to leverage our ideas into collusion resilient CPRFs.

Key-Homomorphic PRFs. In key-homomorphic PRFs, there is a group structure associated with the set of keys, and it is required that for any input x and keys s, t, $F_s(x) + F_t(x) = F_{s+t}(x)$. A construction in the random oracle model was given by Naor, Pinkas and Reingold [22], and the first construction in the standard model was given by Boneh et al. [8] based on the Learning with Errors assumption (LWE), building on a (non key homomorphic) lattice-based PRF of Banerjee, Peikert and Rosen [4]. This was followed by an improved construction by Banerjee and Peikert [3] based on quantitatively better lattice assumptions. The LWE based constructions achieved a slightly weaker notion, namely "almost" key-homomorphism, in which $\|(F_s(x) + F_t(x)) - F_{s+t}(x)\|$ is small, for an appropriately defined norm. This notion is sufficient for the known

applications. Applications of key-homomorphic PRFs include distributed key-distribution, symmetric proxy re-encryption, updatable encryption and PRFs secure against related-key attacks [22,8,19].

Our Results. We view the main contribution of this work as showing how to impose *hidden semantics* into the evaluation process of LWE-based PRFs. Namely, we allow multiple computation paths for computing $F_s(x)$, such that we can selectively block some of these paths based on logic described by a circuit. This is done by extending ideas from the ABE literature, and in particular the ABE scheme of Boneh et al. [7] (see more about this connection below).

It is particularly interesting that previous constructions of PRFs [8,3] can be viewed as a special case of our framework, but ones that only allow a single computational path. Our work therefore highlights that the techniques used for constructing PRFs and for constructing ABE are special cases of the same grand schema. This could hopefully lead to new insights and constructions.

We employ our methods towards presenting a family of (single key secure) constrained key-homomorphic PRFs based on worst-case general lattice assumptions. This is a first step in solving the open problem posed in [9] of achieving (collusion resilient) CPRFs from standard assumptions.

Our construction is selectively secure in the constraint query, namely the adversary needs to decide on the constraint before seeing the public parameters, but is adaptive with regards to PRF oracle queries. We achieve the latter without "complexity leveraging", contrary to [9], and thus we do not require sub-exponential hardness assumptions as they do. This is done by employing our technique of embedding semantics into the evaluation process again. In particular, we embed the semantics of an *admissible hash function*, introduced by Boneh and Boyen [6] into the PRF, which allows us to handle adaptive queries.

Our proofs rely on two closely related hardness assumptions: The Learning with Errors (LWE) assumption, and the one-dimensional Short Integer Solution (1D-SIS) assumption. Both assumptions can be tied to the worst case hardness of general lattice problems such as GapSVP and SIVP, with similar parameters. LWE is sufficient for proving pseudorandomness in the absence of a constrained key. However, once the adversary is given a constrained key, the situation becomes more delicate. In particular, even showing *correctness* in this setting is not straightforward. (Correctness refers to the property that evaluation using the constrained key and using the actual seed result in the same output.) One can show unconditionally that the value computed using the constrained key is *close* (in norm) to the real value of the function but not that they are always equal. A similar issue comes up in the security proof (since the reduction "fabricates" oracle answers in a similar way to the constrained evaluation). Our solution is to use *computational* arguments. Namely to show that it is computationally intractable, under the 1D-SIS assumption, to come up with an input for which the constrained evaluation errs. Therefore even the correctness of our scheme relies on computational assumptions. We note that similar techniques can be

used to strengthen the almost key-homomorphism property into computational key-homomorphism where it is computationally hard to find an input for which key homomorphism does not hold.

The following theorem presents the simplest application of our method, we explain how it can be extended below.

Theorem 1.1. *Let $\mathcal{C}_{\ell,d}$ be the class of size-ℓ depth-d circuits. Then for all polynomials ℓ, d, there exists a $\mathcal{C}_{\ell,d}$-constrained (almost) key-homomorphic family of PRFs without collusion, based on the (appropriately parameterized)* LWE *and* 1D-SIS *assumptions (and hence on the worst-case hardness of appropriately parameterized* GapSVP *and* SIVP *problems).*

Interestingly, we can go beyond bounded size circuits. In fact, we can support any function family with bounded length description, so long as there is a universal evaluator of depth d that takes a function description and an input, and executes the function on the input. Namely, consider a sequence of universal circuits $\{\mathcal{U}_k\}_{k\in\mathbb{N}}$, where $\mathcal{U}_k : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}$. This sequence defines a class of functions $\{0,1\}^* \to \{0,1\}$, where each function F in the class is represented by a string $f \in \{0,1\}^\ell$, and for $x \in \{0,1\}^k$, it holds that $F(x) = \mathcal{U}_k(f,x)$. We call such a function class ℓ-uniform. We are only able to support $\mathcal{U}_k$ whose depth is bounded by some a-priori polynomial in the security parameter d, however in some cases this is sufficient to support all k's that are polynomial in the security parameter. The following theorem states our result with regards to such families.

Theorem 1.2. *Let $\mathcal{C}_{\ell,d}$ be a class of ℓ-uniform functions with depth-d evaluator. Then for all polynomials ℓ, d, there exists a $\mathcal{C}_{\ell,d}$-constrained (almost) key-homomorphic family of PRFs without collusion, based on the (appropriately parameterized)* LWE *and* 1D-SIS *assumptions (and hence on the worst case hardness of appropriately parameterized* GapSVP*,* SIVP*).*

Lastly, we show that the bit-length of the constrained keys in our scheme can be reduced to $\mathrm{poly}(\lambda)$ for some *fixed* polynomial. Namely, completely independent of all of the parameters of the scheme. This is done by using an ABE scheme with short secret keys as a black box. In particular we resort to the same scheme, namely the ABE scheme of Boneh et al. [7], which inspired our constrained PRF construction. This is done by encrypting all of the "components" of the constrained key, and providing them in the public parameters of the construction. Then, the actual constrained key is an ABE secret key which only allows to decrypt the relevant components. We note that this short representation for constrained keys is not homomorphic (however the scheme is still almost key homomorphic with respect to the seed). A theorem statement follows.

Theorem 1.3. *There exists a constrained PRF scheme with the same properties as in Theorem 1.2, and under the same hardness assumptions, where the constrained keys are of asymptotic bit-length* $\mathrm{poly}(\lambda)$*, for an a-priori fixed polynomial.*

See Section 2 for an extended overview of the construction.

Relation to the ABE Construction of Boneh et al. [7]. Our techniques are greatly influenced by the aforementioned LWE-based ABE construction of Boneh et al. [7]. Recall that in ABE, messages are encrypted relative to *attributes* and decryption keys are drawn relative to *functions*. Decryption is possible only if the function f of the decryption key accepts the attribute x of the ciphertext. In order to decrypt a ciphertext, [7] first applies a public procedure that depends on f, x on the ciphertext and then applies the decryption key on the resulting value. Their construction makes sure that for any f, encryptions with regards to all accepting x's will derive a decryptable ciphertext (and all non-accepting x's cannot be decrypted).

Our constrained key for a circuit C is almost identical to an encryption of 0 with attribute C in [7]. The randomness in the encryption roughly corresponds to the seed of the PRF. An application of the PRF on the constrained key includes applying the public procedure of the ABE on the ciphertext, with respect to the function $f = \mathcal{U}$, the universal circuit for the function class to which C belongs. However, there is the question of how to represent the input: We need to be able to evaluate C on any possible input while preserving security. One of our main technical ideas is in showing that this is possible, and in fact can be achieved regardless of the input length. Combined with the framework from [7], we can guarantees that for all x, regardless which C was used to generate the "ciphertext", the output of the public procedure will only depend on x and not on C. The basic idea is therefore to use this value as the PRF value. This does not work as is (for example, it does not imply pseudorandomness for non-accepting x's) and additional ideas are required.

As mentioned above, the PRFs of [8,3] that seem to stem from different ideas and have quite different proofs than [7] can be shown to be special cases of the above paradigm, except f is taken to be an arbitrary formula (a multiplication tree). For details see Section 2.

The novelty in our approach is to show the extra power that is obtained from generalizing these two approaches. We use the universal circuit as a way to embed an undisclosed computation into an LWE instance, and show how to achieve pseudorandomness using tools such as admissible hash functions (which are also embedded into an LWE instance).

Relation with the Constrained PRF of Hofheinz et al. [17]. The work of [17] constructs adaptively secure collusion-resistant CPRFs, namely ones where the challenge x^* needs not be provided ahead of time. Their building blocks are "universal parameters" and *adaptively secure* ABE, which are used as black-box. Note that we achieve adaptive security w.r.t the challenge (but not with respect to the constraint) while relying on techniques which are only known to imply *selectively secure* ABE. Further, whereas [17] use ABE only to implement access control and therefore need to rely on strong assumptions to implement the PRF so as to interface with the ABE, we use ABE techniques to achieve both pseudorandomness and access control. On the flip side, our construction is not collusion resistant, contrary to [17].

Open Problems. The main drawback of our CPRF is its vulnerability to collusion, which severely limits its applicability as a building block. It is an open problem to achieve bounded collusion resilience, even for two constrained keys instead of one and even at the cost of increasing the parameters. Any improvement on this front should be very interesting. Another avenue for research is trying to extend the construction so that there is no restriction on the constraint circuit size, similarly to the multilinear map based construction of [9]. Finally, it would also be interesting to apply this methodology of imposing semantics on a cryptographic computation to other primitives in order to allow more fine-grained access control.

2 Overview of Our Construction

We recall that the LWE assumption asserts that for a uniform vector $\mathbf{s}$ and a matrix $\mathbf{A}$ of appropriate dimensions (over $\mathbb{Z}_q$ for an appropriate q), it holds that $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$, is indistinguishable from uniform, where $\mathbf{e}$ is taken from an appropriate distribution over *low norm* vectors and referred to as the *noise vector*. In this outline we will ignore the generation of $\mathbf{e}^T$ and its evolution during computation process, and just denote it by noise (but of course care will need to be taken in the formal arguments).

The PRF of Banerjee and Peikert [3]. A high-level methodology for constructing PRFs, taken by [8,3] and also in this work, is to take $\mathbf{s}$ as the seed, and to generate for each PRF input x, an LWE matrix $\mathbf{A}_x$ such that the values $\mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$ for the different inputs x are jointly indistinguishable from uniform. Note that almost key homomorphism follows naturally for any implementation of this template, up to the accumulation of noise. The noise issue is handled by taking the PRF value to be a properly scaled down and rounded version of the above, so that the effect of the noise is minimal (and its norm can be bounded below 1). This property is also inherited by our scheme.

As a starting point for deriving our construction, let us revisit the key-homomorphic PRF construction of [3]. Their PRF family was associated with a combinatorial object – a binary tree. Each node v of the tree was associated with an LWE matrix $\mathbf{A}_v$, where the PRF input x determined the matrices for the leaves, and matrices for internal nodes are derived as follows. Given a node v whose children are associated with $\mathbf{A}_l, \mathbf{A}_r$, they define $\mathbf{A}_v = \mathbf{A}_l \cdot \mathbf{G}^{-1}(\mathbf{A}_r)$. In this notation, $\mathbf{G}^{-1}(\cdot)$ is the binary decomposition operator, which breaks each entry in the matrix into the bit vector of length $\log(q)$ of its binary representation. Note that $\mathbf{G}^{-1}(\cdot)$ will always have small norm, and that the inverse operator $\mathbf{G}$, representing binary composition, is linear so it can be represented by a matrix. Thus for all $\mathbf{A}$ it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

Going back to the PRF of [3], the derivation procedure described above allows to associate a matrix with the root of the tree, which depends only on the input x (and on the topology of the tree which is fixed). We will use the root's matrix as our $\mathbf{A}_x$. The proof hinges on the invariant that LWE instances will

be multiplied on the right only by low-norm matrices (of the form $\mathbf{G}^{-1}(\cdot)$), and therefore $\mathbf{s}^T\mathbf{A}_l\mathbf{G}^{-1}(\mathbf{A}_r) + \mathsf{noise} \approx (\mathbf{s}^T\mathbf{A}_l + \mathsf{noise})\mathbf{G}^{-1}(\mathbf{A}_r)$, which allows to replace $(\mathbf{s}^T\mathbf{A}_l + \mathsf{noise})$ with a new uniform vector and propagate to the right.

From Embedded Trees to Embedded Circuits. We show that the operation $\mathbf{A}_v = \mathbf{A}_l \cdot \mathbf{G}^{-1}(\mathbf{A}_r)$ is in fact a special case of a more general operation, inspired by the recent Attribute Based Encryption (ABE) construction of Boneh et al. [7]. We will associate a matrix $\mathbf{A}_v$ *as well as a binary value* x_v with each node, and pay special attention to the matrix $(\mathbf{A}_v - x_v\mathbf{G})$. In particular, considering a node v with children l, r, it holds that

$$(\mathbf{A}_l - x_l\mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{A}_r) + (\mathbf{A}_r - x_r\mathbf{G}) \cdot x_l = \mathbf{A}_l\mathbf{G}^{-1}(\mathbf{A}_r) - x_r x_l \mathbf{G} \ .$$

This generalization associates the semantics of the multiplication operation with the syntactic definition $\mathbf{A}_v = \mathbf{A}_l\mathbf{G}^{-1}(\mathbf{A}_r)$, and it also maintains the invariant that the matrices $(\mathbf{A}_l - x_l\mathbf{G})$ and $(\mathbf{A}_r - x_r\mathbf{G})$ are only multiplied on the right by low norm elements, so that

$$\mathbf{s}^T\Big((\mathbf{A}_l - x_l\mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{A}_r) + (\mathbf{A}_r - x_r\mathbf{G}) \cdot x_l\Big) + \mathsf{noise} \approx \\ \Big(\mathbf{s}^T(\mathbf{A}_l - x_l\mathbf{G}) + \mathsf{noise}\Big) \cdot \mathbf{G}^{-1}(\mathbf{A}_r) + \Big(\mathbf{s}^T(\mathbf{A}_r - x_r\mathbf{G}) + \mathsf{noise}\Big) \cdot x_l \ ,$$

which will play an important role in the security proof. Put explicitly, if the evaluator holds $\mathbf{s}^T(\mathbf{A}_l - x_l\mathbf{G}) + \mathsf{noise}$ and $\mathbf{s}^T(\mathbf{A}_l - x_l\mathbf{G}) + \mathsf{noise}$, then it can compute $\mathbf{s}^T(\mathbf{A}_v - x_l \cdot x_r\mathbf{G}) + \mathsf{noise}$ (and we will obviously define $x_v = x_l \cdot x_r$).

This semantic relation can be extended beyond multiplication gates, and in particular NAND gates can be supported in a fairly similar manner. Furthermore, there is no need to stick to tree structure and one can support arbitrary DAGs, which naturally correspond to circuits. Extending the above postulate, if our DAG corresponds to a circuit C, then having $\mathbf{s}^T(\mathbf{A}_i - x_i\mathbf{G}) + \mathsf{noise}$, for all leaves (= inputs), allows to compute $\mathbf{s}^T(\mathbf{A}_x - C(x)\mathbf{G}) + \mathsf{noise}$. Recalling that the value of the PRF on input x is $\mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$, the aforementioned information allows us to evaluate the PRF at points where $C(x) = 0$. It can also be shown that it is computationally hard to compute the value at points where $C(x) = 1$. We note that this process is practically identical to the public part of the decryption procedure in the [7] ABE (as we explained in Section 1). We also note that since [3] were trying to minimize the complexity of evaluating their PRF, it made no sense in their construction to consider DAGs which only increase the complexity. However, as we show here, there are benefits to embedding a computational process in the PRF evaluation.

Utilizing the Universal Circuit. The tools we describe so far indeed seem to get us closer to our goal of producing constrained keys, but we are still not quite there. What we showed is that for any circuit C, we can devise a PRF with a constrained key for C. Note that we use the negated definition to the one we used before, and allow to evaluate when $C(x) = 0$ and not when $C(x) = 1$. This will be our convention throughout this overview.

In order to reverse the order of quantifiers, we take C to be the universal circuit $\mathcal{U}(F, x)$, and the constrained keys will be of the form $\mathbf{s}^T(\mathbf{A}_i - f_i\mathbf{G}) + \mathsf{noise}$, where the f_i is the ith bit of the description of the constraint F, as well as values for the x wires, which will be of the form $\mathbf{s}^T(\hat{\mathbf{A}}_b - b\mathbf{G}) + \mathsf{noise}$, for both $b \in \{0, 1\}$. These values will allow us to execute F on *any* input x. Note that we can use the same matrices $\hat{\mathbf{A}}_0, \hat{\mathbf{A}}_1$ for all input wires, hence we don't need to commit to the input size when we provide the constrained key.[1] From this description it is obvious why our construction is not collusion resistant: Given two constrained keys for two non identical functions, there exists an i such that the adversary gets both $\mathbf{s}^T\mathbf{A} + \mathsf{noise}$ and $\mathbf{s}^T(\mathbf{A}_i - \mathbf{G}) + \mathsf{noise}$. Recovering $\mathbf{s}^T$ from these values is straightforward and hence all security is lost. Note that for the input values, unlike the function description, we use two different matrices for 0 and 1: $\hat{\mathbf{A}}_0, \hat{\mathbf{A}}_1$, so a similar problem does not occur.

The Problem with Correctness, and a Computational Solution. We introduced two ways to compute the value of the PRF at x: One is to compute $\mathbf{A}_x$ and use the seed $\mathbf{s}^T$ to compute $\mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$, and the other is to use the constrained key to obtain $\mathbf{s}^T(\mathbf{A}_x - F(x)\mathbf{G}) + \mathsf{noise}$, which for $F(x) = 0$ gives $\mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$. The problem is that the *noise value* in these two methods could differ. It is possible to make the difference small by scaling down and rounding, but this is not going to suffice for our purposes (mostly because a similar problem comes up in the security proof). We solve this issue using the 1D-SIS assumption as follows. We first note that the evaluation using the constrained key is essentially evaluation of a linear function with small coefficients on the vectors constituting the constrained key (essentially they get multiplied by bits and by low norm matrices $\mathbf{G}^{-1}(\cdot)$). Secondly, the only way for the two computation paths to not agree is if the value $\mathbf{s}^T\mathbf{A}_x$ is very close to an integer multiple of a number p (which is part of the PRF description). Finally, we notice that by LWE, the vectors in the constrained key are indistinguishable from uniform and independent. Thus, if we encounter such x for which correctness does not work, we can also find a short linear combination of random elements whose scaled down rounded value is close to an integer. In other words, given a uniform vector $\mathbf{v}$ in $\mathbb{Z}_q$, we can find $\mathbf{z}$ such that $\lfloor \langle \mathbf{v}, \mathbf{z} \rangle / p \rfloor$ is "close" to an integer. This is similar to solving a one-dimensional instance of the SIS problem, i.e. $\langle \mathbf{v}, \mathbf{z} \rangle = 0 \pmod{p}$. Indeed, one can show that the 1D-SIS problem is as hard as standard worst-case hard lattice problems via a reduction from [24].

Pseudorandomness and Adaptive Security. Given a constrained key for F, one can compute $\mathbf{s}^T(\mathbf{A}_x - F(x)\mathbf{G}) + \mathsf{noise}$, and indeed if $F(x) = 1$ it is hard to compute $\mathsf{PRF}_{\mathbf{s}}(x) = \mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$. However, we want to argue that this value is *pseudorandom* and furthermore that it remains pseudorandom after adaptive queries to the PRF. Namely, after the adversary sees as many values of the form $\mathsf{PRF}_{\mathbf{s}}(x) = \mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$ as it wishes.

[1] Recall that in [8,3] there are only two matrices altogether. This is sufficient here for the input wires for the same reason, but we need additional matrices to encode the constraint description.

To achieve these goals, we add another feature to the PRF. We consider a new independent LWE matrix $\mathbf{D}$, and define $\mathsf{PRF}_{\mathbf{s}}(x) = \mathbf{s}^T\mathbf{A}_x \cdot \mathbf{G}^{-1}(\mathbf{D}) + \mathsf{noise}$. First of all, we note that given the constrained key, we can still compute the PRF for values where $C(x) = 0$, by first computing $(\mathbf{s}^T\mathbf{A}_x + \mathsf{noise})$ as before, and then multiplying by $\mathbf{G}^{-1}(\mathbf{D})$, which has low norm. However, in general we have

$$\mathsf{PRF}_{\mathbf{s}}(x) \approx \Big(\mathbf{s}^T(\mathbf{A}_x - F(x)\mathbf{G}) + \mathsf{noise}\Big) \cdot \mathbf{G}^{-1}(\mathbf{D}) + F(x)\Big(\mathbf{s}^T\mathbf{D} + \mathsf{noise}\Big) \ ,$$

and it can be shown that for $F(x) = 1$, the second term randomizes the expression, by the LWE assumption.

This handles pseudorandomness for a single query, but not for the case of adaptive queries (since we can only use the pseudorandomness of $(\mathbf{s}^T\mathbf{D} + \mathsf{noise})$ once). To handle adaptive queries we embed semantics into the matrix $\mathbf{D}$ itself. Namely, $\mathbf{D} = \mathbf{D}_x$ will be derived by an application of the universal circuit to the input x and *an admissible hash function* h. Admissible hash functions, introduced by Boneh and Boyen [6], allow (at a very high level) to partition the input space such that with noticeable probability all of the adaptive queries have value $h(x) = 0$, but the challenge query will have $h(x) = 1$. This means that in the proof of security, we can hold a constrained key for h, which will allow us to compute $(\mathbf{s}^T\mathbf{D}_x + \mathsf{noise})$, for all the queries of the adversary, but leave the challenge query unpredictable (to make it pseudorandom, we will multiply in the end by another final $\mathbf{D}'$). This concludes the security argument for adaptive queries.

Key-Homomorphism. As we mention above, key-homomorphism follows since we use the template $\mathsf{PRF}_{\mathbf{s}}(x) = \mathbf{s}^T\mathbf{A}_x + \mathsf{noise}$. We note that the existence of noise means that homomorphism may not be accurate and with some low probability $(\mathsf{PRF}_{\mathbf{s}}(x) + \mathsf{PRF}_{\mathbf{s}'}(x))$ will only be close to $\mathsf{PRF}_{\mathbf{s}+\mathbf{s}'}(x)$ and not identical. However this property is sufficient for many applications.

We point out that our constrained keys are a collection elements of the form $(\mathbf{s}^T\mathbf{A}_i + \mathsf{noise})$, and therefore the scheme is also homomorphic with respect to constrained keys, i.e. constrained keys for the same F w.r.t different keys $\mathbf{s}, \mathbf{s}'$ can be added to obtain a constrained key w.r.t $\mathbf{s} + \mathbf{s}'$.

Reducing the Constrained Key Size. From the above, it follows that the constrained key contains $\ell + 2$ vectors, where ℓ is the bit length of a description of F relative to the universal circuit for the function class. Note that this does not depend directly on the input size to the function. However, indirectly the depth of the universal circuit affects the modulus q that needs to be used.

We show that we can remove the dependence on ℓ altogether using an ABE scheme with short secret keys, such as that of [7]. To do this, we notice that for each constraint function F, the adversary gets either $\mathbf{s}^T\mathbf{A}_i + \mathsf{noise}$ or $\mathbf{s}^T(\mathbf{A}_i - \mathbf{G}) + \mathsf{noise}$, according to the value of the bit f_i. We can prepare for both options by encrypting both vectors using the ABE, each with its own attribute $(i, 0)$ and $(i, 1)$ respectively. All of these encryptions, for all i, will be placed in the public

parameters. Then in order to provide a constrained key, we will provide an ABE secret key for the function that takes (i, b) and returns 0 if and only if $f_i = b$. Given this key, the user can decrypt exactly those vectors that constitute its constrained key. Note that this function can be computed by a depth $O(\log(\ell)) = O(\log(\lambda))$ circuit, and thus the size of the secret key can be made asymptotically independent of all parameters except λ, e.g. by setting the parameters to support depth $\log^2(\lambda)$ circuits.

3 Preliminaries

We first recall some background. For an integer modulus q, let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the ring of integers modulo q. For an integer $p \leq q$, we define the modular "rounding" function

$$\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p \text{ that maps } x \to \lfloor (p/q) \cdot x \rceil$$

and extend it coordinate-wise to matrices and vectors over $\mathbb{Z}_q$. We denote the elements of the standard basis by $\mathbf{u}_1, \mathbf{u}_2, \ldots$, where the dimension will be clear from the context.

We denote distributions (or random variables) that are computationally indistinguishable by $X \overset{c}{\approx} Y$. This refers to the standard notion of negligible distinguishing gap for any polynomial time distinguisher. Our reductions preserve the uniformity of the adversary so by assuming the hardness of our assumption for uniform adversary we get security for our construction against uniform adversaries, and likewise for non-uniform assumptions and adversaries.

The Gadget Matrix. Let $\ell = \lceil \log q \rceil$ and define the "gadget matrix" $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times n\ell}$ where

$$\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^{\ell}$$

We will also refer to this gadget matrix as the "powers-of-two" matrix. We define the inverse function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \to \{0,1\}^{n\ell \times m}$ which expands each entry $a \in \mathbb{Z}_q$ of the input matrix into a column of size ℓ consisting of the bit decomposition of a. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$,

$$\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$$

Norms for Vectors and Matrices. We will always use the infinity norm for vectors and matrices. Namely for a vector $\mathbf{x}$, the norm $\|\mathbf{s}\|$ is the maximal absolute value of an element in $\mathbf{x}$. Similarly, for a matrix $\mathbf{A}$, $\|\mathbf{A}\|$ is the maximal absolute value of any of its entries. If $\mathbf{x}$ is n-dimensional and $\mathbf{A}$ is $n \times m$, then $\left\|\mathbf{x}^T\mathbf{A}\right\| \leq n \cdot \|\mathbf{x}\| \cdot \|\mathbf{A}\|$. We remark that L_1 or L_2 norms can also be used and even achieve somewhat tighter parameters, but the proofs become more complicated.

3.1 Constrained Pseudorandom Function: Definition

In a constrained PRF family [9,10,18], one can compute a constrained PRF key K_C corresponding to any Boolean circuit C. Given K_C, anyone can compute the PRF on inputs x such that $C(x) = 0$. Furthermore, K_C does not reveal any information about the PRF values at the other locations. Below we recall their definition, as given by [9].

Syntax A *constrained* pseudo-random function (PRF) family is defined by a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ where:

- **Key Generation** $\mathsf{KeyGen}(1^\lambda, 1^{k_{\mathsf{in}}}, 1^{k_{\mathsf{out}}})$ is a PPT algorithm that takes as input the security parameter λ, an input length k_{in} and an output length k_{out}, and outputs a PRF key K;
- **Evaluation** $\mathsf{Eval}(K, x)$ is a deterministic algorithm that takes as input a key K, a string $x \in \{0,1\}^{k_{\mathsf{in}}}$ and outputs $y \in \{0,1\}^{k_{\mathsf{out}}}$;
- **Constrained Key Generation** $\mathsf{Constrain}(K, C)$ is a PPT algorithm that takes as input a PRF key K, a circuit $C : \{0,1\}^{k_{\mathsf{in}}} \to \{0,1\}$ and outputs a constrained key K_C;
- **Constrained Evaluation** $\mathsf{ConstrainEval}(K_C, x)$ is a deterministic algorithm that takes as input a constrained key K_C and a string $x \in \{0,1\}^{k_{\mathsf{in}}}$ and outputs either a string $y \in \{0,1\}^{k_{\mathsf{out}}}$ or $\perp$.

We define the notion of *(single key) selective-function* security for constrained PRFs.

Definition 3.1. *A family of PRFs* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ *is a* single-key selective-function constrained PRF *(henceforth, referred to simply as constrained PRF) if it satisfies the following properties:*

- **Functionality computationally preserved under constraining.** *For every* PPT *adversary* (A_0, A_1)*, consider an experiment where we choose* $K \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{k_{\mathsf{in}}}, 1^{k_{\mathsf{out}}})$*,* $(C, \sigma_0) \leftarrow A_0(1^\lambda)$*, and* $K_C \leftarrow \mathsf{Constrain}(K, C)$*. Then:*

$$\Pr\left[\begin{array}{l} \\ x^* \leftarrow A_1^{\mathsf{Eval}(K,\cdot)}(1^\lambda, K_C, \sigma_0); \end{array} \begin{array}{r} C(x^*) = 0 \wedge \\ : \mathsf{Eval}(K, x^*) \neq \mathsf{ConstrainEval}(K_C, x^*) \end{array}\right]$$

is negligible in the security parameter, where C, K, K_C *are selected as described above.*
In words, it is computationally hard to find an x^* *such that* $C(x^*) = 0$*, and yet the result of the constrained evaluation differs from the actual PRF evaluation.*

- **Pseudorandom at constrained points.** *For every* PPT *adversary* (A_0, A_1, A_2)*, consider an experiment where* $K \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{k_{\mathsf{in}}}, 1^{k_{\mathsf{out}}})$*,* $(C, \sigma_0) \leftarrow A_0(1^\lambda)$*, and* $K_C \leftarrow \mathsf{Constrain}(K, C)$*. Then:*

$$\Pr\left[\begin{array}{l} b \leftarrow \{0,1\}; \\ (x^*, \sigma_1) \leftarrow A_1^{\mathsf{Eval}(K,\cdot)}(1^\lambda, K_C, \sigma_0); \\ \texttt{If } b = 0,\ y^* = \mathsf{Eval}(K, x^*), \\ \texttt{Else} \quad\ y^* \leftarrow \{0,1\}^{k_{\mathsf{out}}} \end{array} \begin{array}{r} \\ : C(x^*) = 1 \wedge \\ A_2(1^\lambda, y^*, \sigma_1) = b \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

The correctness and security properties could potentially be combined into one game, but we choose to present them as two distinct properties for the sake of clarity.

3.2 Learning with Errors

The Learning with Errors (LWE) problem was introduced by Regev [25] as a generalization of "learning parity with noise" [5,2]. We now define the decisional version of LWE. (Unless otherwise stated, we will treat all vectors as column vectors in this paper).

Definition 3.2 (Decisional LWE (DLWE) [25]). *Let λ be the security parameter, $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$ be integers and $\chi = \chi(\lambda)$ be a probability distribution over $\mathbb{Z}$. The $\mathrm{DLWE}_{n,q,\chi}$ problem states that for all $m = \mathrm{poly}(n)$, letting $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$, the following distributions are computationally indistinguishable:*

$$\left(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T\right) \stackrel{c}{\approx} \left(\mathbf{A}, \mathbf{u}^T\right)$$

There are known quantum (Regev [25]) and classical (Peikert [23]) reductions between $\mathrm{DLWE}_{n,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be a discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$ for some $\alpha < 1$. We write $\mathrm{DLWE}_{n,q,\alpha}$ to indicate this instantiation. We now state a corollary of the results of [25,23,20,21]. These results also extend to additional forms of q (see [20,21]).

Corollary 3.3 ([25,23,20,21]). *Let $q = q(n) \in \mathbb{N}$ be either a prime power $q = p^r$, or a product of co-prime numbers $q = \prod q_i$ such that for all i, $q_i = \mathrm{poly}(n)$, and let $\alpha \geq \sqrt{n}/q$. If there is an efficient algorithm that solves the (average-case) $\mathrm{DLWE}_{n,q,\alpha}$ problem, then:*

- *There is an efficient quantum algorithm that solves $\mathsf{GapSVP}_{\widetilde{O}(n/\alpha)}$ (and $\mathsf{SIVP}_{\widetilde{O}(n/\alpha)}$) on any n-dimensional lattice.*
- *If in addition $q \geq \tilde{O}(2^{n/2})$, there is an efficient classical algorithm for $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$ on any n-dimensional lattice.*

Recall that GapSVP_γ is the (promise) problem of distinguishing, given a basis for a lattice and a parameter d, between the case where the lattice has a vector shorter than d, and the case where the lattice doesn't have any vector shorter than $\gamma \cdot d$. SIVP is the search problem of finding a set of "short" vectors. The best known algorithms for GapSVP_γ ([27]) require at least $2^{\tilde{\Omega}(n/\log\gamma)}$ time. We refer the reader to [25,23] for more information.

In this work, we will only consider the case where $q \leq 2^n$. Furthermore, the underlying security parameter λ is assumed to be polynomially related to the dimension n.

3.3 One-Dimensional Short Integer Solution (SIS) and Variants

We present a special case of the well known Short Integer Solution (SIS) problem [1].

Definition 3.4. *The One-Dimensional Short Integer Solution problem, denoted* 1D-SIS$_{q,m,t}$*, is the following problem. Given a uniformly distributed vector* $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^m$*, find* $\mathbf{z} \in \mathbb{Z}^m$ *such that* $\|\mathbf{z}\| \leq t$ *and also* $\langle \mathbf{v}, \mathbf{z} \rangle \in [-t, t] + q\mathbb{Z}$*.*

For appropriately chosen moduli q, the 1D-SIS$_{q,m,t}$ problem is as hard as worst-case lattice problems. This follows from the techniques in the classical worst-case to average-case reduction of Ajtai [1]. We state below the version due to Regev [24].

Corollary 3.5 (Section 4 in [24] and Proposition 4.7 in [13]). *Let* $n \in \mathbb{N}$ *and* $q = \prod_{i \in n} p_i$*, where all* $p_1 < p_2 < \ldots < p_n$ *are co-prime. Let* $m \geq c \cdot n \log q$ *(for some universal constant c). Assuming that* $p_1 \geq t \cdot \omega(\sqrt{mn \log n})$*, the one-dimensional* SIS *problem* 1D-SIS$_{q,m,t}$ *is at least as hard as* $\mathsf{SIVP}_{t \cdot \widetilde{O}(\sqrt{mn})}$ *and* $\mathsf{GapSVP}_{t \cdot \widetilde{O}(\sqrt{mn})}$*.*

Proof. The hardness of a closely related problem is established by combining the techniques in [24, Section 4] and [13, Proposition 4.7]: Given $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^{m+1}$, find $\mathbf{y}$ with $\|\mathbf{y}\| \leq t$ such that $\langle \mathbf{a}, \mathbf{y} \rangle = 0 \pmod q$.

We now show how to convert an instance for this problem into an instance of 1D-SIS. Given an instance $\mathbf{a} \in \mathbb{Z}_q^{m+1}$, we consider the first component a_1. If this element is not a unit (i.e. invertible) in $\mathbb{Z}_q$, then the reduction aborts. Otherwise it defines $\mathbf{v} = a_1^{-1} \cdot [a_2, \ldots, a_{m+1}]$. Given a solution $\mathbf{z}$ for 1D-SIS on input $\mathbf{v}$, we define $\mathbf{y}$ by letting $\mathbf{y} = [-\langle \mathbf{v}, \mathbf{z} \rangle, x_1, \ldots, x_m]$. It is easy to verify that $\langle \mathbf{a}, \mathbf{y} \rangle = a_1 \cdot (-\langle \mathbf{v}, \mathbf{z} \rangle + \langle \mathbf{v}, \mathbf{z} \rangle) = 0 \pmod q$. Further, by definition, $\|\mathbf{y}\| \leq t$.

Next, we define a related problem which will be useful for our reductions.

Definition 3.6. *Let* $q = p \cdot \prod_{i \in n} p_i$*, where all* $p_1 < p_2 < \ldots < p_n$ *are all co-prime and co-prime with* p *as well. Further let* $m \in \mathbb{N}$*. The* 1D-SIS-R$_{q,p,t,m}$ *problem is the following: Given* $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^m$*, find* $\mathbf{z} \in \mathbb{Z}^m$ *with* $\|\mathbf{z}\| \leq t$ *such that* $\langle \mathbf{v}, \mathbf{z} \rangle \in [-t, t] + (q/p)\mathbb{Z}$*.*

The following corollary establishes the hardness of 1D-SIS-R based on 1D-SIS.

Corollary 3.7. *Let* q, p, t, m *be as in Definition 3.6. Then* 1D-SIS-R$_{q,p,t,m}$ *is at least as hards as* 1D-SIS$_{q/p,t,m}$*.*

Proof. The reduction works in the obvious way: Given an input $\mathbf{v} \in \mathbb{Z}_{q/p}^m$ for 1D-SIS$_{q/p,t,m}$, we embed $\mathbf{v}$ in $\mathbf{v}' \in \mathbb{Z}_q^m$, using CRT representation. Namely $\mathbf{v}' = \mathbf{v} \pmod{q/p}$ and $\mathbf{v}' = \mathbf{r} \pmod p$, where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^m$. Then given a solution $\mathbf{z}$ for 1D-SIS-R$_{q,p,t,m}$ with input $\mathbf{v}'$, we claim that $\mathbf{z}$ is also a solution for 1D-SIS$_{q/p,t,m}$ with input $\mathbf{v}$. This follows since by definition $\|\mathbf{z}\| \leq t$, and since $\langle \mathbf{v}, \mathbf{z} \rangle \equiv \langle \mathbf{v}', \mathbf{z} \rangle \pmod{q/p}$.

3.4 Admissible Hash Functions

The concept of admissible hash functions was defined by Boneh and Boyen [6] to convert selectively secure identity based encryption (IBE) schemes into fully secure ones. In this paper, we use admissible hash functions for our PRF construction. Our definition of admissible hash functions below will follow that of Cash, Hofheinz, Kiltz and Peikert [12] with minor changes (in particular, note that we do not require that the bad set is efficiently recognizable).

Definition 3.8 ([6,12]). *Let $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$ be a family of hash functions such that $\mathcal{H}_\lambda \subseteq \left(\{0,1\}^* \to \{0,1\}^\ell\right)$ for some $\ell = \ell(\lambda)$. We say that $\mathcal{H}$ is a family of admissible hash functions if for every $H \in \mathcal{H}$ there exists a set bad_H of "bad string-tuples" such that the following two properties hold:*

1. *For every PPT algorithm $\mathcal{A}$, there is a negligible function ν such that*

$$\Pr[(x^{(0)}, \ldots, x^{(t)}) \in \mathsf{bad}_H \mid H \leftarrow \mathcal{H}_\lambda, (x^{(0)}, \ldots, x^{(t)}) \leftarrow \mathcal{A}(1^\lambda, H)] \leq \nu(\lambda)$$

   where the probability is over the choice of $H \leftarrow \mathcal{H}_\lambda$ and the coins of $\mathcal{A}$.
2. *Let $\mathcal{L} = \{0,1\}^{2\ell}$, and for all $L \in \mathcal{L}$ define $\Pi_L : \{0,1\}^\ell \to \{0,1\}$ to be the string comparison with wildcards function. Namely, write L as a pair of strings $(\alpha, \beta) \in \{0,1\}^\ell$, and define*

$$\Pi_{L=(\alpha,\beta)}(w) = 1 \Leftrightarrow \forall i \in [\ell] \left((\alpha_i = 0) \vee (\beta_i = w_i)\right) .$$

   Intuitively, Π is a string comparison function with wildcards. It compares w and β only at those points where $\alpha_i = 1$. Note that this representation is somewhat redundant but it will be useful for our application.
   Then, we require that for every polynomial $t = t(\lambda)$ there exists a noticeable function $\Delta_t(\lambda)$ and an efficiently sampleable distribution $\mathcal{L}_t$ over $\mathcal{L}$ such that for every $H \in \mathcal{H}_\lambda$ and sequences $(x^{(0)}, \ldots, x^{(t)}) \notin \mathsf{bad}_H$ with $x^{(0)} \notin \{x^{(1)}, \ldots, x^{(t)}\}$, we have:

$$\Pr_{L \leftarrow \mathcal{L}_t}[\Pi_L(H(x^{(0)})) \wedge \overline{\Pi_L(H(x^{(1)}))} \wedge \cdots \wedge \overline{\Pi_L(H(x^{(t)}))}] \geq \Delta_t(\lambda)$$

It has been shown by [6] that a family of admissible hash functions can be constructed based on any collision resistant hash function. In particular one can instantiate it based on the SIS problem (for virtually any parameter setting for which the problem is hard), which is at least as hard as LWE. Therefore throughout this manuscript we assume the existence of an LWE-based family of admissible hash functions, which will not add an additional assumption to our construction.

3.5 Attribute-Based Encryption

We define (leveled) attribute-based encryption, following [16,15]. An attribute-based encryption scheme for a class of predicate circuits $\mathcal{C}$ (namely, circuits with a single bit output) consists of four algorithms ($\mathcal{ABE}.\mathsf{Setup}$, $\mathcal{ABE}.\mathsf{KeyGen}$, $\mathcal{ABE}.\mathsf{Enc}$, $\mathcal{ABE}.\mathsf{Dec}$).

$\mathcal{ABE}.\mathsf{Setup}(1^\lambda, 1^\ell, 1^d) \to (\mathsf{pp}, \mathsf{msk})$: The setup algorithm gets as input the security parameter λ, the length ℓ of the attributes and the maximum depth of the predicate circuits d, and outputs the public parameter $(\mathsf{pp}, \mathsf{mpk})$, and the master key msk. All the other algorithms get pp as part of their input.

$\mathcal{ABE}.\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$: The key generation algorithm gets as input msk and a predicate specified by $C \in \mathcal{C}$ (of depth at most d). It outputs a secret key (C, sk_C).

$\mathcal{ABE}.\mathsf{Enc}(\mathsf{pp}, \mathbf{x}, m) \to \mathsf{ct}$: The encryption algorithm gets as input mpk, attributes $\mathbf{x} \in \{0,1\}^\ell$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $(\mathbf{x}, \mathsf{ct})$.

$\mathcal{ABE}.\mathsf{Dec}((C, \mathsf{sk}_C), (\mathbf{x}, \mathsf{ct})) \to m$: The decryption algorithm gets as input a circuit C and the associated secret key sk_C, attributes $\mathbf{x}$ and an associated ciphertext ct, and outputs either $\perp$ or a message $m \in \mathcal{M}$.

Correctness. We require that for all ℓ, d, all $(\mathbf{x}, C)$ such that $\mathbf{x} \in \{0,1\}^\ell$, C has depth at most d and $C(\mathbf{x}) = 1$, for all $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathcal{ABE}.\mathsf{Setup}(1^\lambda, 1^\ell, 1^d)$, all $\mathsf{sk}_C \leftarrow \mathcal{ABE}.\mathsf{KeyGen}(\mathsf{msk}, C)$, all $\mathsf{ct} \leftarrow \mathcal{ABE}.\mathsf{Enc}(\mathsf{pp}, \mathbf{x}, m)$, and all $m \in \mathcal{M}$,

$$\mathsf{Dec}((C, \mathsf{sk}_C), (\mathbf{x}, \mathsf{ct})) = m) \ .$$

Security Definition. We define selective security of ABE, which is sufficient for our purposes. We allow the adversary to make multiple challenge message queries, which is equivalent to the single query case but will be easier for us to work with.

Definition 3.9. *For a stateful adversary $\mathcal{A}$, we define the advantage function $\mathsf{Adv}^{ABE}_{\mathcal{A}}$ to be*

$$\Pr\left[b = b' : \begin{array}{l} b \xleftarrow{\$} \{0,1\}; \\ \mathbf{x}_1, \ldots, \mathbf{x}_Q \leftarrow \mathcal{A}(1^\lambda, 1^\ell, 1^d); \\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathcal{ABE}.\mathsf{Setup}(1^\lambda, 1^\ell, 1^d); \\ \{(m_{0,i}, m_{1,i})\}_{i \in [Q]} \leftarrow \mathcal{A}^{\mathcal{ABE}.\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{pp}), \forall i. |m_{0,i}| = |m_{1,i}|; \\ \mathsf{ct}_i \leftarrow \mathcal{ABE}.\mathsf{Enc}(\mathsf{pp}, \mathbf{x}_i, m_{b,i}); \\ b' \leftarrow \mathcal{A}^{\mathcal{ABE}.\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}_1, \ldots, \mathsf{ct}_Q) \end{array} \right] - \frac{1}{2}$$

with the restriction that all queries C that $\mathcal{A}$ makes to $\mathcal{ABE}.\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ satisfies $C(\mathbf{x}_i) = 0$ for all i (that is, sk_C does not decrypt the ciphertext corresponding to any of the $\mathbf{x}_i$). An attribute-based encryption scheme is selectively secure *if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{ABE}_{\mathcal{A}}$ is a negligible function in λ.*

We will use a special type of attribute-based encryption scheme with succinct keys, namely one where $|\mathsf{sk}_C|$ does not grow with the size of the circuit C, but rather only its depth.

Theorem 3.10 ([7]). *Let λ be the security parameter, and $d \in \mathbb{N}$. Let $n = n(\lambda, d)$, $q = q(\lambda, d) = n^{O(d)}$, and let χ be a $\mathsf{poly}(n)$-bounded error distribution. Then, there is a selectively secure ABE scheme for the class of depth-d-bounded circuits, based on the hardness of $\mathrm{DLWE}_{n,q,\chi}$. Furthermore, the secret key sk_C for a circuit C has size $\mathsf{poly}(\lambda, n, d)$.*

4 Embedding Circuits into Matrices

In this section, we present the core techniques that we use in our construction. In essence, we use a method, developed in a recent work by Boneh et al. [7] to "embed" bits $x_1, \ldots, x_k$ into matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$ and compute a circuit F on these matrices. This is done through a pair of algorithms $(\mathsf{ComputeA}, \mathsf{ComputeC})$ satisfying the following properties:

1. The deterministic algorithm $\mathsf{ComputeA}$ takes as input a circuit $F : \{0,1\}^k \to \{0,1\}$ and k matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$, and outputs a matrix $\mathbf{A}_F$; and
2. The deterministic algorithm $\mathsf{ComputeC}$ takes as input a bit string $\mathbf{x} = (x_1, \ldots, x_k) \in \{0,1\}^k$, and k LWE samples $\mathbf{s}^T(\mathbf{A}_i + x_i\mathbf{G}) + \mathbf{e}_i$, and outputs an LWE sample $\mathbf{s}^T(\mathbf{A}_F + F(\mathbf{x}) \cdot \mathbf{G}) + \mathbf{e}_F$ associated to the output matrix $\mathbf{A}_F$ and the output bit $F(\mathbf{x})$.

These algorithms are closely modeled on the work of Boneh et al. [7]. We now describe how these algorithms work, and what their properties are.

The Algorithm $\mathsf{ComputeA}$. Given a circuit F, input matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$ (corresponding to the k input wires) and an auxiliary matrix $\mathbf{A}_0$, the $\mathsf{ComputeA}$ procedure works inductively, going through the gates of the circuit F from the input to the output. Assume without loss of generality that the circuit F is composed of NOT and AND gates. For every AND gate $g = (u, v; w)$, assume inductively that we have computed matrices $\mathbf{A}_u$ and $\mathbf{A}_v$ for the input wires u and v. Define

$$\mathbf{A}_w = -\mathbf{A}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v)$$

For every NOT gate $g = (u; w)$, define

$$\mathbf{A}_w = \mathbf{A}_0 - \mathbf{A}_u$$

The Algorithm $\mathsf{ComputeC}$. Given a circuit F, an input $x \in \{0,1\}^k$ and LWE samples $(\mathbf{A}_i, \mathbf{y}_i)$, the $\mathsf{ComputeC}$ algorithm works as follows. For each AND gate $g = (u, v; w)$, assume that we have computed LWE samples $(\mathbf{A}_u, \mathbf{y}_u)$ and $(\mathbf{A}_v, \mathbf{y}_v)$ for the input wires u and v. Define

$$\mathbf{y}_w = x_u \cdot \mathbf{y}_v - \mathbf{y}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v)$$

where x_u and x_v are the bits on wires u and v when evaluating the circuit F on input x. For every NOT gate $g = (u; w)$, define

$$\mathbf{y}_w = \mathbf{y}_0 - \mathbf{y}_u$$

We will need the following lemma about the behavior of $\mathsf{ComputeA}$ and $\mathsf{ComputeC}$. (We remind the reader that we use $||\cdot||$ to denote the ℓ_∞ norm).

Lemma 4.1. *Let F be a depth-d Boolean circuit on k input bits, and let $x \in \{0,1\}^k$ be an input. Let $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_k \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{y}_0, \ldots, \mathbf{y}_k \in \mathbb{Z}_q^m$ be such that*

$$||\mathbf{y}_i - \mathbf{s}^T(\mathbf{A}_i + x_i\mathbf{G})|| \leq B \qquad \text{for } i = 0, 1, \ldots, k.$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$ *and* $B = B(\lambda)$. *Let* $\mathbf{A}_F \leftarrow \mathsf{ComputeA}(F, \mathbf{A}_0, \ldots, \mathbf{A}_k)$ *and* $\mathbf{y}_F \leftarrow \mathsf{ComputeC}(F, x, \mathbf{A}_0, \ldots, \mathbf{A}_k, \mathbf{y}_0, \ldots, \mathbf{y}_k)$. *Then,* $||\mathbf{y}_F - \mathbf{s}^T(\mathbf{A}_F + F(\mathbf{x}) \cdot \mathbf{G})|| \leq m^{O(d)} \cdot B$.

Furthermore, $\mathbf{y}_F$ *is a "low-norm" linear function of* $\mathbf{y}_0, \ldots, \mathbf{y}_k$. *That is, there are matrices* $\mathbf{Z}_0, \ldots, \mathbf{Z}_k$ *(which depend on the function* F*, the input* $\mathbf{x}$*, and the input matrices* $\mathbf{A}_0, \ldots, \mathbf{A}_k$*) such that* $\mathbf{y}_F = \sum_{i=0}^{k} \mathbf{y}_i \mathbf{Z}_i$ *and* $||\mathbf{Z}_i|| \leq m^{O(d)} \cdot B$.

Proof. We show this by induction on the levels of the circuit F, starting from the input. Consider two cases.

AND gate. Consider an AND gate $g = (u, v; w)$ where the input wires are at level L, and assume that $\mathbf{y}_u = \mathbf{s}^T(\mathbf{A}_u + x_u\mathbf{G}) + \mathbf{e}_u$ and $\mathbf{y}_v = \mathbf{s}^T(\mathbf{A}_v + x_v\mathbf{G}) + \mathbf{e}_v$, with $||\mathbf{e}_u||, ||\mathbf{e}_v|| \leq (m+1)^L \cdot B$. Now,

$$\begin{aligned}
\mathbf{y}_w &= x_u \cdot \mathbf{y}_v - \mathbf{y}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v) \\
&= x_u \cdot \left(\mathbf{s}^T(\mathbf{A}_v + x_v\mathbf{G}) + \mathbf{e}_v\right) - \left(\mathbf{s}^T(\mathbf{A}_u + x_u\mathbf{G}) + \mathbf{e}_u\right) \cdot \mathbf{G}^{-1}(\mathbf{A}_v) \\
&= \mathbf{s}^T\left(x_u\mathbf{A}_v + x_u x_v\mathbf{G} - \mathbf{A}_u\mathbf{G}^{-1}(\mathbf{A}_v) - x_u\mathbf{A}_v\right) + \left(-\mathbf{e}_u\mathbf{G}^{-1}(\mathbf{A}_v) + x_u\mathbf{e}_v\right) \\
&= \mathbf{s}^T(\mathbf{A}_w + x_w\mathbf{G}) + \mathbf{e}_w
\end{aligned}$$

where $\mathbf{A}_w = -\mathbf{A}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v)$, $x_w = x_u x_v$, and

$$||\mathbf{e}_w|| \leq m \cdot ||\mathbf{e}_u|| + ||\mathbf{e}_v|| \leq (m+1) \cdot (m+1)^L \cdot B \leq (m+1)^{L+1} \cdot B$$

NOT gate. In a similar vein, for a NOT gate $g = (u; w)$, assume that $\mathbf{y}_u = \mathbf{s}^T(\mathbf{A}_u + x_u\mathbf{G}) + \mathbf{e}_u$, with $||\mathbf{e}_u|| \leq (m+1)^L \cdot B$. Then,

$$\begin{aligned}
\mathbf{y}_w = \mathbf{y}_0 - \mathbf{y}_u &= \mathbf{s}^T(\mathbf{A}_0 + \mathbf{G} - \mathbf{A}_u - x_u\mathbf{G}) + (\mathbf{e}_0 - \mathbf{e}_u) \\
&= \mathbf{s}^T(\mathbf{A}_w + (1 - x_u)\mathbf{G}) + \mathbf{e}_w
\end{aligned}$$

where $\mathbf{A}_w = \mathbf{A}_0 - \mathbf{A}_u$, $x_w = 1 - x_u$, and

$$||\mathbf{e}_w|| \leq ||\mathbf{e}_0|| + ||\mathbf{e}_u|| \leq B + (m+1)^L \cdot B \leq (m+1)^{L+1} \cdot B$$

Thus, $\mathbf{y}_F = \mathbf{s}^T\mathbf{A}_F + \mathbf{e}_F$ where $||\mathbf{e}_F|| \leq m^{O(d)} \cdot B$. Furthermore, both transformations are linear functions on $\mathbf{y}_u$ and $\mathbf{y}_v$, as required.

5 Constrained PRF

5.1 Construction

A family of functions $\mathcal{F} \subseteq (\{0,1\}^* \to \{0,1\})$ is z-uniform if each function $F \in \mathcal{F}$ can be described by a string in $\{0,1\}^z$ (we associate F with its description), and there exists a uniform circuit family $\{\mathcal{U}_k\}_{k \in \mathbb{N}}$ such that $\mathcal{U}_k : \{0,1\}^z \times \{0,1\}^k \to$

$\{0,1\}$ such that for all $x \in \{0,1\}^k$ it holds that $\mathcal{U}_k(F,x) = F(x)$. We assume for the sake of simplicity that the depth of $\mathcal{U}_k$ grows monotonically with k and for all d we let k_d to be the maximal input size for which $\mathcal{U}_k$ has depth at most d. We define $\mathcal{F}_d$ to be such that $F \in \mathcal{F}$ is undefined for inputs of length $k > k_d$. We call such a family d-depth-bounded.

Our constrained PRF for a z-uniform d-depth-bounded family $\mathcal{F}$ works as follows.

- $\mathsf{KeyGen}(1^\lambda, 1^z, 1^d)$: The key generation algorithm takes as input the maximum size z and depth d of the constraining circuits. Let $\mathcal{H}$ be a family of admissible hash functions (see Section 3.4) and let $\ell = \ell(\lambda)$ be the output length of hash functions in the family.
  Let $n = n(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ be parameters chosen as described in Section 5.2 below, let $m = n\lceil \log q \rceil$.
  Generate $z + 2\ell + 3$ matrices as follows: let $\mathbf{A}_0$ and $\mathbf{A}_1$ be the "input matrices", let $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_z$ be the "function matrices", let $\mathbf{C}_1, \ldots, \mathbf{C}_{2\ell}$ be the "partitioning matrices", and let $\mathbf{D}$ be an "auxiliary matrix". All of these matrices are uniform in $\mathbb{Z}_q^{n \times m}$ (note that the "gadget matrix" $\mathbf{G}$ has the same dimensions). In addition sample an admissible hash function $H \xleftarrow{\$} \mathcal{H}_\lambda$. The public parameters consist of

$$\mathcal{PP} = (H, \mathbf{A}_0, \mathbf{A}_1, \mathbf{B}_1, \ldots, \mathbf{B}_z, \mathbf{C}_1, \ldots, \mathbf{C}_{2\ell}, \mathbf{D})$$

  The seed of the PRF is a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$.
- $\mathsf{Eval}(\mathbf{s}, \mathcal{PP}, \mathbf{x})$ takes as input the PRF seed $\mathbf{s}$, the public parameters $\mathcal{PP}$, and an input $x \in \{0,1\}^k$ such that $k \le k_d$ (i.e. $\mathcal{U}_k$ is of depth $\le d$), and works as follows.
  Recall that $\mathcal{U}_k : \{0,1\}^z \times \{0,1\}^k \to \{0,1\}$ is the universal circuit that takes a description of a function F and an input x and outputs $\mathcal{U}_k(F,x) = F(x)$. Let $\Pi : \{0,1\}^{2\ell} \times \{0,1\}^\ell \to \{0,1\}$ denote the circuit that computes $\Pi(L,w) = \Pi_L(w)$ from Definition 3.8. Note that Π can be implemented by a binary circuit of depth $\log(\ell) + O(1)$.
  Let $(x_1, \ldots, x_k)$ denote the bits of x. Let $w = H(x)$, and let $w_1, \ldots, w_\ell$ be its bits. Compute

$$\mathbf{B}_{\mathcal{U}} \leftarrow \mathsf{ComputeA}\big(\mathcal{U}_k, \mathbf{B}_1, \ldots, \mathbf{B}_z, \mathbf{A}_{x_1}, \mathbf{A}_{x_2}, \ldots, \mathbf{A}_{x_k}\big) \tag{1}$$

$$\mathbf{C}_\Pi \leftarrow \mathsf{ComputeA}\big(\Pi, \mathbf{C}_1, \ldots, \mathbf{C}_{2\ell}, \mathbf{A}_{w_1}, \mathbf{A}_{w_2}, \ldots, \mathbf{A}_{w_\ell}\big) \tag{2}$$

  and output

$$\mathsf{PRF}_{\mathbf{s}}(\mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{B}_{\mathcal{U}} \cdot \mathbf{G}^{-1}(\mathbf{C}_\Pi) \cdot \mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p$$

- $\mathsf{Constrain}(\mathbf{s}, \mathcal{PP}, F)$ takes as input the PRF key $\mathbf{s}$ and a circuit F (of size at most z) and does the following. Compute

$$\mathbf{a}_b = \mathbf{s}^T(\mathbf{A}_b + b \cdot \mathbf{G}) + \mathbf{e}_{1,b}^T \in \mathbb{Z}_q^m \text{ for } b \in \{0,1\}$$
$$\mathbf{b}_i = \mathbf{s}^T(\mathbf{B}_i + f_i \cdot \mathbf{G}) + \mathbf{e}_{2,i}^T \in \mathbb{Z}_q^m \text{ for all } i \in [z]$$

where the vectors $\mathbf{e}$ are drawn from an error distribution χ to be specified later (in Section 5.2).

The constrained seed K_F is the tuple $\left(\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_1, \ldots, \mathbf{b}_z\right) \in (\mathbb{Z}_q^m)^{z+2}$.

– $\mathsf{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})$ takes as input the constrained key K_F and an input $\mathbf{x}$. It computes

$$\mathbf{b}_{\mathcal{U},\mathbf{x}} \leftarrow \mathsf{ComputeC}\Big(\mathcal{U}, (\mathbf{b}_1, \ldots, \mathbf{b}_z, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_k}), (f_1, \ldots, f_z, x_1, \ldots, x_k)\Big)$$

and outputs $\left\lfloor \mathbf{b}_{\mathcal{U},\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi}) \cdot \mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p$, where $\mathbf{C}_{\Pi}$ is defined as above.

5.2 Setting the Parameters

Let us start by providing a typical parameter setting, and then explain how parameters can be modified and the effect on security.

Consider setting $n(\lambda, d) = (\lambda \cdot d)^c$, for a constant c that will be discussed shortly. We will set χ to be a discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$ s.t. $\alpha q = \Theta(\sqrt{n})$. We define $n' = \lambda$ and let $p_1, \ldots, p_{n'} = m^{O(d+\log \ell)}$ be all primes, and $p = \mathrm{poly}(\lambda)$ (in fact, there is a lot of freedom in the choice of p, and it can be as large as $m^{O(d+\log \ell)}$ under the same asymptotic hardness). Finally, let $q = p \cdot (\alpha q) \cdot \prod_{i \in [n']} p_i = m^{n' \cdot O(d+\log \ell)} = 2^{\widetilde{O}(\lambda \cdot d)} = 2^{\widetilde{O}(n^{1/c})}$ (recall that $\ell = \mathrm{poly}(\lambda)$).

This parameter setting translates into a PRF with $m = n \lceil \log q \rceil \cdot \Theta(\log \lambda)$ output bits per input, whose security is based (as we show in the next section) on the hardness of approximating lattice problems to within a factor of $2^{\widetilde{O}(n^{1/c})}$.

Taking larger values of c will increase the hardness of the underlying lattice problem, but at the cost of considerably increasing the element sizes.

5.3 Security

Throughout this section, we let $\mathcal{F}$ be a family of z-uniform functions and let d be a depth bound (both can depend on λ). We let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. We let $\mathcal{H}$ be the family of admissible hash functions as described in Section 3.4, with range $\{0,1\}^{\ell}$.

Theorem 5.1. *Let $\mathcal{F}$ be a family of z-uniform functions and let d be a depth bound (both can depend on λ). Let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. Further let $m' = m \cdot (z + 2\ell + 3)$, and $\gamma = \omega(\sqrt{n \log \lambda}) \cdot p \cdot m^{O(d+\log \ell)}$. Assuming the hardness of* $\mathrm{DLWE}_{n,q,\chi}$, $\text{1D-SIS-R}_{q,p,\gamma,m'}$ *and the admissible hash function family $\mathcal{H}$, the scheme $\mathcal{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ is a single-key secure selective-function secure constrained PRF for $\mathcal{F}$.*

We note that the hardness of all three assumptions translates to the worst case hardness of approximating lattice problems such as GapSVP and SIVP to within sub-exponential factors.

Proof. Let $\mathcal{A}$ be a PPT selective-constraint adaptive-input adversary against $\mathcal{CPRF}_{z,d}$. Let $t = \text{poly}(\lambda)$ be the (polynomial) number of input queries made by $\mathcal{A}$ (w.l.o.g). Let ϵ be the advantage of $\mathcal{A}$ in the constrained PRF game. We let $B = \alpha q \cdot \omega(\sqrt{\log \lambda})$. It holds that with all but negligible probabilities, all samples that we take from χ will have absolute value at most B. For the duration of the proof we assume that this is indeed the case.

The proof will proceed by a sequence of hybrids (or experiments) where the challenger samples a bit $b \in \{0,1\}$ and interacts with $\mathcal{A}$. We let $\text{Adv}_{\mathsf{H}}(\mathcal{A})$ denote the probability that $\mathcal{A}$ outputs b in hybrid H.

Hybrid H_0. This hybrid is the legitimate constrained PRF security game. The challenger generates $(\mathbf{s}, \mathcal{PP}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^z, 1^d)$. It gets $F \in \{0,1\}^z$ from $\mathcal{A}$ and produces a constrained key $K_F \leftarrow \mathsf{Constrain}(\mathbf{s}, \mathcal{PP}, F)$. It then sends $\mathcal{PP}, K_F$ to $\mathcal{A}$. At this point $\mathcal{A}$ adaptively makes queries $x^{(i)} \in \{0,1\}^*$, and the challenger computes $y^{(i)} \leftarrow \mathsf{Eval}(\mathbf{s}, \mathcal{PP}, x^{(i)})$ and returns it to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $x^* \in \{0,1\}^*$. If $b = 0$ then the challenger returns $y^* \leftarrow \mathsf{Eval}(\mathbf{s}, \mathcal{PP}, x^*)$, and if $b = 1$ it returns a random y^*. Therefore, we have

$$\text{Adv}_{\mathsf{H}_0}(\mathcal{A}) \geq 1/2 + \epsilon \ .$$

Hybrid H_1. This is the notorious "artificial abort" phase. Let $\Delta_t = \Delta_t(\lambda)$ be the noticeable function from Definition 3.8. This hybrid is identical to the previous one, except in the last step the challenger flips a coin and with probability $1 - \Delta_t/2$ aborts the experiment (hence giving the adversary no information on b).

The adversary's advantage thus degrades appropriately:

$$\text{Adv}_{\mathsf{H}_1}(\mathcal{A}) \geq (\Delta_t/2) \cdot (1/2 + \epsilon) + (1 - \Delta_t/2) \cdot (1/2) = 1/2 + \epsilon \cdot \Delta_t/2 \ .$$

Hybrid H_2. In this hybrid, we associate some meaning with the artificial abort. Intuitively, the abort will be associated with a failure of the admissible hash function to partition the queries correctly. We are guaranteed that correct partitioning happens with probability $\geq \Delta_t$ (except for sequences that are hard to generate), but we would like to make it *(almost) exactly* $\Delta_t/2$ so as to not correlate the adversary's success probability with the string L (the loss of the 2 factor is due to probability estimation).

Specifically, in this hybrid, rather than flipping a coin at the end of the experiment, the challenger does the following. For all $\vec{x} = (x^{(1)}, \ldots, x^{(t)}, x^*)$, we define the event $\mathsf{GoodPartition}_{L,\vec{x}}$ to be the event in which $\Pi_L(H(x^{(1)})) = \cdots = \Pi_L(H(x^{(t)})) = 0$ and $\Pi_L(H(x^*)) = 1$, and define $\delta_{\vec{x}} = \Pr_{L \xleftarrow{\$} \mathcal{L}_t}[\mathsf{GoodPartition}_{\vec{x},L}]$. The challenger will first compute an estimate $\tilde{\delta}_{\vec{x}}$ of $\delta_{\vec{x}}$ by sampling multiple values of L from $\mathcal{L}_t$ and using Chernoff (both additive and multiplicative). Using $\text{poly}(\lambda)$-many samples we can compute $\tilde{\delta}_{\vec{x}}$ such that

$$\Pr\left[\left|\delta_{\vec{x}} - \tilde{\delta}_{\vec{x}}\right| > \Delta_t/4\right] \leq 2^{-\lambda} \ .$$

and in addition if $\delta_{\vec{x}} \geq \Delta_t/2$ then

$$\Pr\left[\left|\frac{\delta_{\vec{x}}}{\tilde{\delta}_{\vec{x}}} - 1\right| > \epsilon/2\right] \leq 2^{-\lambda} .$$

The challenger will then perform as follows: (i) It first verifies that $\tilde{\delta}_{\vec{x}} \geq \frac{3}{4}\Delta_t$, and aborts if this is not the case. (ii) It then samples $L \xleftarrow{\$} \mathcal{L}_t$ and aborts if $\mathsf{GoodPartition}_{\vec{x},L}$ did not occur (note that by our definitions above, this happens with probability $1 - \delta_{\vec{x}}$ over the choice of L). (iii) Then it flips a coin with probability $\frac{\tilde{\delta}_{\vec{x}} - \Delta_t/2}{\tilde{\delta}_{\vec{x}}}$ and aborts if the outcome is 1. Otherwise it carries out the experiment towards completion.

To analyze the effect on the success probability, we first notice that the probability that $\tilde{\delta}_{\vec{x}} < \frac{3}{4}\Delta_t$ (abortion is step (i)) is negligible. This is since, except with $2^{-\lambda}$ probability, this indicates that $\delta_{\vec{x}} < \Delta_t$, which implies that $\vec{x} \in \mathsf{bad}_H$. Definition 3.8 guarantees that this happens with probability at most $\nu(\lambda) = \mathrm{negl}(\lambda)$.

If the above abort did not occur, we know that $\delta_{\vec{x}} \geq \Delta_t/2$ (except with probability $2^{-\lambda}$), we first notice that the total probability of abort in steps $(ii) + (iii)$

$$1 - \delta_{\vec{x}} + \delta_{\vec{x}} \cdot \frac{\tilde{\delta}_{\vec{x}} - \Delta_t/2}{\tilde{\delta}_{\vec{x}}} = 1 - \frac{\delta_{\vec{x}}}{\tilde{\delta}_{\vec{x}}}\Delta_t/2 \in \left[(1 - \Delta_t/2) - \epsilon\Delta_t/4,\ (1 - \Delta_t/2) + \epsilon\Delta_t/4\right]$$

It therefore follows that if there was no abort in step (i), then the adversary's view in H_2 is within statistical distance $2^{-\lambda} + \epsilon\Delta_t/4$ from its view in H_1.

Putting all steps together, we get that

$$\mathrm{Adv}_{\mathsf{H}_2}(\mathcal{A}) \geq 1/2 + \epsilon \cdot \Delta_t/2 - \nu(\lambda) - O(2^{-\lambda}) - \epsilon\Delta_t/4 = 1/2 + \epsilon \cdot \Delta_t/4 - \mathrm{negl}(\lambda) .$$

Hybrid H_3. In this hybrid, the challenger first samples $L \xleftarrow{\$} \mathcal{L}_t$, and then, for each $x^{(i)}$ in turn, it checks whether $\Pi_L(H(x^{(i)})) = 0$, and immediately aborts if not. Similarly, upon receiving x^*, it checks whether $\Pi_L(H(x^*)) = 1$ and immediately aborts if not. Otherwise it continues the same as H_2.

It is rather straightforward to see that the $\mathcal{A}$'s advantage does not change. The cases in which we abort are exactly the same as the ones in the previous hybrid (since it is sufficient that a single $x^{(i)}$ does not give the required value in order to abort). Further, the sampling of L has been completely independent of all the other randomness in the experiment so it might as well happen in the beginning. We conclude that

$$\mathrm{Adv}_{\mathsf{H}_3}(\mathcal{A}) = \mathrm{Adv}_{\mathsf{H}_2}(\mathcal{A}) \geq 1/2 + \epsilon \cdot \Delta_t/4 - \mathrm{negl}(\lambda) .$$

Hybrid H_4. In this hybrid, the challenger changes the way the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are generated. Recall that our security game is constraint-selective, namely $\mathcal{A}$ produces the constraint F before seeing the public parameters.

Therefore, here, the challenger waits until receiving F from $\mathcal{A}$ and only generates the public parameters at that point (note that by then L has also been specified). To generate the public parameters, the matrix $\mathbf{D}$ is produced identically

to before. In addition, the challenger samples matrices $\{\hat{\mathbf{A}}_\beta\}_{\beta\in\{0,1\}}$, $\{\hat{\mathbf{B}}_i\}_{i\in[z]}$, $\{\hat{\mathbf{C}}_i\}_{i\in[2\ell]}$ It then sets

$$\begin{aligned}\mathbf{A}_\beta &= \hat{\mathbf{A}}_\beta - \beta\mathbf{G}\\ \mathbf{B}_i &= \hat{\mathbf{B}}_i - f_i\mathbf{G}\\ \mathbf{C}_i &= \hat{\mathbf{C}}_i - L_i\mathbf{G}\end{aligned}$$

The remainder of the experiment remains unchanged.

Since the distributions of the $\mathbf{A}, \mathbf{B}, \mathbf{C}$ matrices is identical to their original uniform distributions, it follows that

$$\mathrm{Adv}_{\mathsf{H}_4}(\mathcal{A}) = \mathrm{Adv}_{\mathsf{H}_3}(\mathcal{A}) \ .$$

Hybrid H_5. In this hybrid, the adversary changes the way it computes the outputs $y^{(i)}$. Recall that $K_F = (\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_1, \ldots, \mathbf{b}_z)$ is the constrained key given to $\mathcal{A}$. Let us denote

$$\begin{aligned}\mathbf{c}_i &= \mathbf{s}^T(\mathbf{C}_i + L_i\mathbf{G}) + \mathbf{e}_{3,i}^T \quad \text{for all } i \in [z]\\ \mathbf{d} &= \mathbf{s}^T\mathbf{D} + \mathbf{e}_4^T\end{aligned}$$

where $\mathbf{e}_{3,i}$ are sampled coordinate-wise from χ, and $\mathbf{e}_4$ is sampled coordinate-wise from χ'.

In this hybrid, in order to answer input queries, the challenger first computes

$$\mathbf{b}_{\mathcal{U},x^{(i)}} \leftarrow \mathsf{ComputeC}\Big(\mathcal{U}, (\mathbf{b}_1, \ldots, \mathbf{b}_z, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_k}), (f_1, \ldots, f_z, x_1^{(i)}, \ldots, x_k^{(i)})\Big)$$

and then, letting $w^{(i)} = H(x^{(i)})$

$$\mathbf{c}_{\Pi,w^{(i)}} \leftarrow \mathsf{ComputeC}\Big(\Pi, (\mathbf{c}_1, \ldots, \mathbf{c}_{2\ell}, \mathbf{a}_{w_1}, \ldots, \mathbf{a}_{w_\ell}), (L_1, \ldots, L_{2\ell}, w_1^{(i)}, \ldots, w_\ell^{(i)})\Big)$$

We recall that by Lemma 4.1 it holds that

$$\begin{aligned}\mathbf{b}_{\mathcal{U},x^{(i)}}^T &= \mathbf{s}^T(\mathbf{B}_{\mathcal{U},x^{(i)}} + F(x^{(i)})\cdot\mathbf{G}) + \mathbf{e}_{\mathcal{U}}^T\\ \mathbf{c}_{\Pi,w^{(i)}}^T &= \mathbf{s}^T(\mathbf{C}_{\Pi,w^{(i)}} + \Pi_L(w^{(i)})\cdot\mathbf{G}) + \mathbf{e}_{\Pi}^T \ ,\end{aligned}$$

for some $\mathbf{e}_{\mathcal{U}}, \mathbf{e}_{\Pi}$ for which $\|\mathbf{e}_{\mathcal{U}}\| \le B \cdot m^{O(d)}$, $\|\mathbf{e}_{\Pi}\| \le B \cdot m^{O(\log \ell)}$.

We recall that by definition

$$\begin{aligned}
\mathsf{PRF}_{\mathbf{s}}(x^{(i)}) &= \left\lfloor \mathbf{s}^T \mathbf{B}_{\mathcal{U},x^{(i)}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p \\
&= \Big\lfloor \mathbf{s}^T (\mathbf{B}_{\mathcal{U},x^{(i)}} + F(x^{(i)})\mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \\
&\qquad\qquad - F(x^{(i)}) \mathbf{s}^T \mathbf{C}_{\Pi,w^{(i)}} \mathbf{G}^{-1}(\mathbf{D}) \Big\rceil_p \\
&= \Big\lfloor \mathbf{s}^T (\mathbf{B}_{\mathcal{U},x^{(i)}} + F(x^{(i)})\mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \\
&\quad - F(x^{(i)}) \mathbf{s}^T (\mathbf{C}_{\Pi,w^{(i)}} + \Pi_L(w^{(i)})\mathbf{G}) \mathbf{G}^{-1}(\mathbf{D}) \\
&\qquad + F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{s}^T \mathbf{D} \Big\rceil_p \\
&= \Big\lfloor \mathbf{b}^T_{\mathcal{U},x^{(i)}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \mathbf{c}^T_{\Pi,w^{(i)}} \mathbf{G}^{-1}(\mathbf{D}) \\
&\quad + F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{d}^T + \mathbf{e}'^T \Big\rceil_p \ , \qquad (3)
\end{aligned}$$

where

$$\mathbf{e}'^T = -\mathbf{e}^T_{\mathcal{U}} \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) + F(x^{(i)}) \mathbf{e}^T_{\Pi} \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{e}^T_4 \qquad (4)$$

which implies that $\|\mathbf{e}'\| \leq E$ for some $E = (m^{O(d)} + m^{O(\log \ell)}) \cdot B$.

To analyze the distinguishing probability between these hybrids, for any input x (and $w = H(x)$) we define the event $\mathsf{Borderline}_x$ as the event where there exists $j \in [m]$ such that:

$$\begin{aligned}
(\mathbf{b}^T_{\mathcal{U},x} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w}) \cdot \mathbf{G}^{-1}(\mathbf{D}) - F(x) \cdot \mathbf{c}^T_{\Pi,w} \cdot \mathbf{G}^{-1}(\mathbf{D}) \\
+ F(x) \cdot \Pi_L(w) \cdot \mathbf{d}^T) \cdot \mathbf{u}_j \in [-E, E] + (q/p)\mathbb{Z} \ ,
\end{aligned}$$

where we recall that $\mathbf{u}_j$ is the jth indicator vector. Namely, this is the probability that one of the coordinates of the vector $\mathbf{b}^T_{\mathcal{U},x} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w}) \mathbf{G}^{-1}(\mathbf{D}) - F(x) \mathbf{c}^T_{\Pi,w} \mathbf{G}^{-1}(\mathbf{D}) + F(x) \Pi_L(w) \mathbf{d}^T$ is "dangerously close" to being rounded in the wrong direction.

By definition of rounding, if $\neg \mathsf{Borderline}_{x^{(i)}}$, then

$$\begin{aligned}
\mathsf{PRF}_{\mathbf{s}}(x^{(i)}) = \lfloor \mathbf{b}^T_{\mathcal{U},x^{(i)}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \mathbf{c}^T_{\Pi,w^{(i)}} \mathbf{G}^{-1}(\mathbf{D}) \\
+ F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{d}^T \rceil_p \ .
\end{aligned}$$

The challenger in this hybrid, given a query $x^{(i)}$, will first check whether $\mathsf{Borderline}_{x^{(i)}}$. If the event happens, the challenger aborts. Otherwise it returns $\mathsf{PRF}_{\mathbf{s}}(x^{(i)})$ as defined above. Note that the challenger only needs to respond to queries $x^{(i)}$ for which $\Pi_L(w^{(i)}) = \Pi_L(H(x^{(i)})) = 0$, which do not depend on $\mathbf{d}$, a fact that will be important later on.

Finally, on the challenge query x^*, unless abort is needed, it holds that $F(x^*) = 1$ and $\Pi_L(w^*) = 1$ (where $w^* = H(x^*)$) and therefore, unless the event $\mathsf{Borderline}_{x^*}$ happens, it holds that

$$\mathsf{PRF}_{\mathbf{s}}(x^*) = \left\lfloor \mathbf{b}^T_{\mathcal{U},x^*} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}})\mathbf{G}^{-1}(\mathbf{D}) - \mathbf{c}^T_{\Pi,w^*}\mathbf{G}^{-1}(\mathbf{D}) + \mathbf{d}^T \right\rceil_p \ .$$

The challenger will therefore abort if $\mathsf{Borderline}_{x^*}$ and return the aforementioned value otherwise (that is if the bit b is 0; if $b = 1$ then of course a uniform value is returned).

It follows that if we define $\mathsf{Borderline} = (\vee_i \mathsf{Borderline}_{x^{(i)}}) \vee \mathsf{Borderline}_{x^*}$, then

$$|\mathrm{Adv}_{\mathsf{H}_5}(\mathcal{A}) - \mathrm{Adv}_{\mathsf{H}_4}(\mathcal{A})| \leq \Pr_{\mathsf{H}_5}[\mathsf{Borderline}] \ .$$

We will bound $\Pr_{\mathsf{H}_5}[\mathsf{Borderline}]$ as a part of our analysis in the next hybrid.

As a final remark on this hybrid, we note that in order to execute this hybrid, the challenger does not need to access $\mathbf{s}$ itself, but rather only the $\mathbf{a}_\beta, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}$ vectors. This will be useful in the next hybrid.

Hybrid H_6. In this hybrid, all $\mathbf{a}_\beta, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}$ are sampled from the uniform distribution. Everything else remains the same. We note that by definition, in hybrid H_5:

$$\begin{aligned}
\mathbf{a}^T_\beta &= \mathbf{s}^T \hat{\mathbf{A}}_\beta + \mathbf{e}^T_{1,\beta} \\
\mathbf{b}^T_i &= \mathbf{s}^T \hat{\mathbf{B}}_i + \mathbf{e}^T_{2,i} \\
\mathbf{c}^T_i &= \mathbf{s}^T \hat{\mathbf{C}}_i + \mathbf{e}^T_{3,i} \\
\mathbf{d}^T &= \mathbf{s}^T \mathbf{D} + \mathbf{e}^T_4 \ ,
\end{aligned}$$

where all $\hat{\mathbf{A}}_\beta, \hat{\mathbf{B}}_i, \hat{\mathbf{C}}_i, \mathbf{D}$ are uniformly distributed, and all $\mathbf{e}^T_{1,\beta}, \mathbf{e}^T_{2,i}, \mathbf{e}^T_{3,i}, \mathbf{e}^T_4$ are sampled coordinate-wise from χ. The $\mathrm{DLWE}_{n,q,\chi}$ assumption therefore asserts that:

$$|\mathrm{Adv}_{\mathsf{H}_6}(\mathcal{A}) - \mathrm{Adv}_{\mathsf{H}_5}(\mathcal{A})| \leq \mathrm{negl}(\lambda) \ .$$

Furthermore, since $\mathsf{Borderline}$ is an efficiently recognizable event, it also holds that

$$\left| \Pr_{\mathsf{H}_6}[\mathsf{Borderline}] - \Pr_{\mathsf{H}_5}[\mathsf{Borderline}] \right| = \mathrm{negl}(\lambda) \ . \tag{5}$$

In H_6, the probability of $\mathsf{Borderline}$ can be bounded under the 1D-SIS-R assumption.

Claim. Under the $\text{1D-SIS-R}_{q,p,\gamma,m'}$ assumption, it holds that $\Pr_{\mathsf{H}_6}[\mathsf{Borderline}] = \mathrm{negl}(\lambda)$, where $m' = m \cdot (2 + z + 2\ell + 1)$, and $\gamma = p \cdot B \cdot m^{O(d + \log \ell)}$.

Proof. Let $\mathbf{v} \in \mathbb{Z}_q^{(2+z+2\ell+1)m}$ be an input to $\text{1D-SIS-R}_{q,p,\gamma,m'}$. Then define $\mathbf{a}_\beta, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}$ be so that their concatenation is $\mathbf{v}$.

The reduction executes H_6 as the challenger, using the vectors defined above. We claim that if Borderline occurs, then we solve 1D-SIS-R. This follows since if Borderline occurs then we found x, j such that

$$(\mathbf{b}_{\mathcal{U},x}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w})\mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)})\mathbf{c}_{\Pi,w}^T\mathbf{G}^{-1}(\mathbf{D}) + F(x)\Pi_L(w)\mathbf{d}^T)\mathbf{u}_j \in [-E, E] + (q/p)\mathbb{Z} \ .$$

However, by Lemma 4.1, it follows that

$$\mathbf{b}_{\mathcal{U},x}^T = \sum_{\beta\in\{0,1\}} \mathbf{a}_\beta^T\mathbf{R}'_{1,\beta} + \sum_{i\in[z]} \mathbf{b}_i^T\mathbf{R}'_{2,i}$$
$$\mathbf{c}_{\Pi,x}^T = \sum_{\beta\in\{0,1\}} \mathbf{a}_\beta^T\mathbf{R}''_{1,\beta} + \sum_{i\in[2\ell]} \mathbf{c}_i^T\mathbf{R}''_{3,i}$$

where $\left\|\mathbf{R}'_{1,\beta}\right\|, \left\|\mathbf{R}'_{2,i}\right\| \leq m^{O(d)}$ and $\left\|\mathbf{R}''_{1,\beta}\right\|, \left\|\mathbf{R}''_{3,i}\right\| \leq m^{O(\log \ell)}$. It follows that there exists an (efficiently derivable) matrix $\mathbf{R}_0$ such that

$$\mathbf{b}_{\mathcal{U},x}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w})\mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)})\mathbf{c}_{\Pi,w}^T\mathbf{G}^{-1}(\mathbf{D}) + F(x)\Pi_L(w)\mathbf{d}^T = \mathbf{v}^T\mathbf{R}_0 \ ,$$

and $\|\mathbf{R}_0\| \leq m^{O(d+\log \ell)}$.

Finally,

$$\langle \mathbf{v}, \mathbf{R}_0 \cdot \mathbf{u}_j \rangle \in [-E, E] + (q/p)\mathbb{Z} \ ,$$

with $\|\mathbf{R}_0 \cdot \mathbf{u}_j\| \leq \|\mathbf{R}_0\| \leq m^{O(d+\log \ell)}$ and $E = B \cdot m^{O(d+\log \ell)} = m^{O(d+\log \ell)}$. Thus $\mathbf{R}_0 \cdot \mathbf{u}_j$ is a valid solution for 1D-SIS-$\mathrm{R}_{q,p,\gamma,m'}$. The claim thus follows.

Putting together Claim 6 and Eq. (5), we get that

$$\Pr_{\mathsf{H}_5}[\mathsf{Borderline}] \leq \Pr_{\mathsf{H}_6}[\mathsf{Borderline}] + \mathrm{negl}(\lambda) \leq \mathrm{negl}(\lambda) \ .$$

and thus, finally

$$\left|\mathrm{Adv}_{\mathsf{H}_5}(\mathcal{A}) - \mathrm{Adv}_{\mathsf{H}_6}(\mathcal{A})\right| \leq \mathrm{negl}(\lambda) \ .$$

Finally, we notice that the vector $\mathbf{d}$ is only used when answering the challenge query in the case of $b = 0$. This means that in the adversary's view, the answer it gets when $b = 0$ is uniform and independent of its view so far, exactly the same as the case $b = 1$ where an actual random vector is returned. It follows that

$$\mathrm{Adv}_{\mathsf{H}_6}(\mathcal{A}) = 1/2 \ .$$

On the other hand

$$\mathrm{Adv}_{\mathsf{H}_6}(\mathcal{A}) \geq 1/2 + \epsilon\Delta_t/4 - \mathrm{negl}(\lambda) \ ,$$

and thus

$$\epsilon \leq \frac{\mathrm{negl}(\lambda)}{\Delta_t/4} = \mathrm{negl}(\lambda) \ .$$

It follows that $\mathcal{A}$ cannot achieve a noticeable advantage in the constrained PRF experiment under the $\mathrm{DLWE}_{q,n,\chi}$ assumption.

5.4 Computational Functionality Preserving

We now prove the computational functionality preservation of our scheme, as per Definition 3.1. Throughout this section, we let $\mathcal{F}$ be a family of z-uniform functions and let d be a depth bound (both can depend on λ). We let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. We let $\mathcal{H}$ be the family of admissible hash functions as described in Section 3.4, with range $\{0,1\}^\ell$.

Theorem 5.2. *Let $\mathcal{F}$ be a family of z-uniform functions and let d be a depth bound (both can depend on λ). Let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. Further let $m' = m \cdot (z + 2\ell + 3)$, and $\gamma = \omega(\sqrt{n \log \lambda}) \cdot p \cdot m^{O(d+\log \ell)}$.*

Assuming the hardness of $\mathrm{DLWE}_{n,q,\chi}$ *and* $\text{1D-SIS-R}_{q,p,\gamma,m'}$*, the scheme $\mathcal{CPRF}$ is computationally functionality preserving.*

We note that the hardness of both assumptions translates to the worst case hardness of approximating lattice problems such as GapSVP and SIVP to within sub-exponential factors.

Proof (outline). The theorem follows from an argument practically identical to that made in Hybrids $\mathsf{H}_5, \mathsf{H}_6$ of the proof of Theorem 5.1.

Recall that we showed that $\mathsf{Borderline}$ events only happen with negligible probability, and therefore with all but negligible probability, it holds that the PRF value at point $x^{(i)}$ is exactly equal to

$$\left\lfloor \mathbf{b}^T_{\mathcal{U},x^{(i)}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}})\mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)})\mathbf{c}^T_{\Pi,w^{(i)}}\mathbf{G}^{-1}(\mathbf{D}) + F(x^{(i)})\Pi_L(w^{(i)})\mathbf{d}^T \right\rceil_p .$$

However, when $F(x^{(i)}) = 0$, this term simplifies to

$$\left\lfloor \mathbf{b}^T_{\mathcal{U},x^{(i)}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}})\mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p$$

which is exactly $\mathsf{ConstrainEval}(K_F, \mathcal{PP}, x^{(i)})$ by definition. Functionality is thus preserved with all but negligible probability.

5.5 Other Properties

We describe several other properties that our construction satisfies.

Unconditional Almost-Correctness. We have shown that our constrained PRF satisfies a computational correctness property, namely that it is hard to find an input $\mathbf{x}$ such that $\mathsf{PRF}_K(\mathbf{x}) \neq \mathsf{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})$. We are also able to show unconditionally that the constrained evaluation and the actual PRF evaluation do not differ by much, for any input $\mathbf{x}$. Indeed, by Equation 3 and 4, we have

$$||\mathsf{PRF}_K(\mathbf{x}) - \mathsf{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})||_\infty \leq m^{O(d)} \cdot B$$

Key Homomorphism. Our PRF is also "almost key homomorphic" in the sense that $\mathsf{PRF}_{\mathbf{s}}(\mathbf{x}) + \mathsf{PRF}_{\mathbf{s}'}(\mathbf{x})$ is close to $\mathsf{PRF}_{\mathbf{s}+\mathbf{s}'}(\mathbf{x})$ for any keys $\mathbf{s}$ and $\mathbf{s}'$ and any input $\mathbf{x}$. Recall that our PRF is

$$\mathsf{PRF}_{\mathbf{s}}(\mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{B}_{\mathcal{U}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi}) \cdot \mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p$$

For any keys $\mathbf{s}_i$ and input $\mathbf{x}$, denoting $\mathbf{s}_i^T \mathbf{B}_{\mathcal{U}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi}) \cdot \mathbf{G}^{-1}(\mathbf{D})$ as $\mathbf{h}_i$, we have

$$||\mathsf{PRF}_{\sum \mathbf{s}_i}(\mathbf{x}) - \sum \mathsf{PRF}_{\mathbf{s}_i}(\mathbf{x})||_\infty = \left|\left| \left\lfloor \sum_i \mathbf{h}_i \right\rceil_p - \sum_i \lfloor \mathbf{h}_i \rceil_p \right|\right|_\infty \leq k+1$$

Constrained-Key Homomorphism. Our constrained keys are "almost homomorphic" as well, in the same sense as above. That is, if K_F and K'_F are constrained versions of PRF keys K and K' for the same function F, the summation $K_F + K'_F$ is a constrained version of $K + K'$ for the function F. For any input $\mathbf{x}$, we then have that $\mathsf{ConstrainEval}(K_F + K'_F, \mathcal{PP}, \mathbf{x})$ is close to $\mathsf{PRF}_{K+K'}(\mathbf{x})$.

We remark that techniques similar to what we used in showing computational correctness can be used to strengthen the almost key-homomorphism property into computational key-homomorphism where it is computationally hard to find an input for which key homomorphism does not hold.

6 Succinct Constrained Keys

In this section we show how to reduce the size of the constrained key so that asymptotically it depends only on the security parameter and independent of the function class. The construction builds upon the scheme $\mathcal{CPRF}$ from Section 5 but reduces the key size by utilizing an attribute based encryption scheme (ABE). In particular, the constrained keys in our new system have size $\mathsf{poly}(\lambda)$, independent of the parameters of the constraining circuit (namely, its size or depth).

Our *succinct* constrained PRF $\mathcal{SCPRF}$ for a z-uniform d-depth-bounded family $\mathcal{F}$ works as follows.

- $\mathsf{KeyGen}(1^\lambda, 1^z, 1^d)$: The key generation algorithm takes as input the maximum size z and depth d of the constraining circuits. Let $t = O(\log z)$ to be specified later.
  It starts by calling $\mathcal{CPRF}.\mathsf{KeyGen}(1^\lambda, 1^z, 1^d)$ to obtain the seed $\mathbf{s}$, and public parameters $\mathcal{PP} = (H, \mathbf{A}_0, \mathbf{A}_1, \{\mathbf{B}_i\}_{i\in[z]})$.
  It then generates: $\mathbf{a}_\beta = \mathbf{s}^T(\mathbf{A}_\beta + \beta\mathbf{G}) + \mathbf{e}_{1,\beta}^T$ and $\mathbf{b}_{i,\beta} = \mathbf{s}^T(\mathbf{B}_i + \beta\mathbf{G}) + \mathbf{e}_{2,i,\beta}^T$. Note that any possible constrained key of $\mathcal{CPRF}$ consists of $\mathbf{a}_0$ and $\mathbf{a}_1$, together with a subset of $\{\mathbf{b}_{i,\beta}\}_{i\in[z],\beta\in\{0,1\}}$.
  Next it generates parameters for the ABE scheme $(\mathcal{ABE}.\mathsf{msk}, \mathcal{ABE}.\mathsf{pp}) \leftarrow \mathcal{ABE}.\mathsf{Setup}(1^\lambda, 1^t)$, and generates $\mathsf{ct}_{i,\beta} \leftarrow \mathcal{ABE}.\mathsf{Enc}(\mathcal{ABE}.\mathsf{pp}, (i,\beta), \mathbf{b}_{i,\beta})$, encryptions with (i,β) as the "attributes" and $\mathbf{b}_{i,\beta}$ as the "message".
  The public parameters consist of

$$\mathcal{SCPRF}.\mathcal{PP} = (\mathcal{CPRF}.\mathcal{PP}, \mathcal{ABE}.\mathcal{PP}, \mathbf{a}_0, \mathbf{a}_1, \{\mathsf{ct}_{i,\beta}\}_{i,\beta})$$

The seed for $\mathcal{SCPRF}$ contains a seed for $\mathcal{CPRF}$, namely a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$, and in addition the ABE master secret key $\mathcal{ABE}.\mathsf{msk}$. We note that in fact $\mathbf{s}$ can be retrieved from the public parameters using $\mathcal{ABE}.\mathsf{msk}$ and therefore it is not necessary to give it explicitly. However, it is more natural to think of $\mathbf{s}$ as a part of the seed. In particular $\mathbf{s}$ will be used to evaluate $\mathcal{SCPRF}$ (see Eval below) and $\mathcal{ABE}.\mathsf{msk}$ will be used to produce constrained keys (see $\mathsf{Constrain}$ below).

- $\mathsf{Eval}(\mathbf{s}, \mathcal{PP}, \mathbf{x})$ takes as input the PRF seed $\mathbf{s}$, the public parameters $\mathcal{PP}$ which contains $\mathcal{CPRF}.\mathsf{pp}$, and an input $x \in \{0,1\}^k$ such that $k \leq k_d$ (i.e. $\mathcal{U}_k$ is of depth $\leq d$), and outputs the result of the CPRF evaluation, namely $\mathcal{CPRF}.\mathsf{Eval}(\mathbf{s}, \mathcal{CPRF}.\mathsf{pp}, \mathbf{x})$.
- $\mathsf{Constrain}(\mathcal{ABE}.\mathsf{msk}, F)$ takes as input the ABE master secret key $\mathcal{ABE}.\mathsf{msk}$ and a circuit F (represented as a string in $\{0,1\}^z$) and does the following. Consider the function:

$$\phi_F(i, \beta) = \begin{cases} 1, \text{ if } F_i = \beta \\ 0, \text{ otherwise} \end{cases}$$

  Note that ϕ_F can be computed by a depth $O(\log z)$ circuit (whose depth is independent of the depth of F itself), the parameter t from above is set to be equal to this depth. We recall Section 3.5
  The constrained key for F is the ABE token for ϕ_F, namely

$$K_F = \mathcal{ABE}.\mathsf{KeyGen}(\mathcal{ABE}.\mathsf{msk}, \phi_F)$$

- $\mathsf{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})$ takes as input the constrained key K_F, the public parameters $\mathcal{PP}$ and an input $\mathbf{x}$.
  Recalling that $\mathcal{PP} = (\mathcal{CPRF}.\mathsf{pp}, \mathcal{ABE}.\mathsf{pp}, \mathbf{a}_0, \mathbf{a}_1, \{\mathsf{ct}_{i,\beta}\})$, and that K_F is the ABE decryption key for the function ϕ_F, it first decrypts to obtain $\mathbf{b}_i = \mathcal{ABE}.\mathsf{Dec}(K_F, \mathsf{ct}_{i,F_i})$, and then applies the constrained evaluation algorithm $\mathcal{CPRF}.\mathsf{ConstrainEval}\big((\mathbf{a}_0, \mathbf{a}_1, \{\mathbf{b}_i\}), \mathcal{CPRF}.\mathcal{PP}, \mathbf{x}\big)$.

The correctness follows in a straightforward manner from the correctness of $\mathcal{ABE}$ and $\mathcal{CPRF}$. The constrained key size of $\mathcal{SCPRF}$ is derived from that of $\mathcal{ABE}$ and is $\mathrm{poly}(\lambda, t) = \mathrm{poly}(\lambda, \log z)$. It follows that there exists a $\mathrm{poly}(\lambda)$ asymptotic upper bound on the key sizes that applies for all polynomial values of z. Security is stated in the following theorem, the proof can be found in the extended version [11].

Theorem 6.1. *If $\mathcal{CPRF}$ is a single-key secure constrained pseudorandom function for function class $\mathcal{F}$ (Definition 3.1), which is built according to the template in Section 5, and if $\mathcal{ABE}$ is a selectively secure ABE scheme (Definition 3.9), then the scheme $\mathcal{SCPRF}$ described above is a secure single-key CPRF for $\mathcal{F}$.*

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, pp. 99–108. ACM (1996)

2. Alekhnovich, M.: More on average case vs approximation complexity. In: Proceedings of 44th Symposium on Foundations of Computer Science (FOCS 2003), Cambridge, MA, USA,, October 11-14, pp. 298–307. IEEE Computer Society (2003)
3. Banerjee, A., Peikert, C.: New and improved key-homomorphic pseudorandom functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 353–370. Springer, Heidelberg (2014)
4. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)
5. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994)
6. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
7. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014)
8. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic prfs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013)
9. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
10. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
11. Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic prfs from standard lattice assumptions or: How to secretly embed a circuit in your prf. Cryptology ePrint Archive, Report 2015/032 (2015), `http://eprint.iacr.org/`
12. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. J. Cryptology 25(4), 601–639 (2012)
13. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. Electronic Colloquium on Computational Complexity (ECCC) 14(133) (2007)
14. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986); Extended abstract in FOCS 84
15. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) Symposium on Theory of Computing Conference, STOC 2013, Palo Alto, CA, USA, June 1-4, pp. 545–554. ACM (2013)
16. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) Proceedings of the 13th ACM Conference on Computer and Communications Security CCS 2006, October 30 - November 3, pp. 89–98. ACM (2006)
17. Hofheinz, D., Kamath, A., Koppula, V., Waters, B.: Adaptively secure constrained pseudorandom functions. In: Cryptology ePrint Archive, Report 2014/720 (2014), `http://eprint.iacr.org/`

18. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A., Gligor, V.D., Yung, M. (eds.) 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, Berlin, Germany, November 4-8, pp. 669–684. ACM (2013)
19. Lewi, K., Montgomery, H.W., Raghunathan, A.: Improved constructions of prfs secure against related-key attacks. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 44–61. Springer, Heidelberg (2014)
20. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011)
21. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)
22. Naor, M., Pinkas, B., Reingold, O.: Distributed pseudo-random functions and kdcs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 327–346. Springer, Heidelberg (1999)
23. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 2009, May 31 - June 2, pp. 333–342 (2009)
24. Regev, O.: Lattices in computer science - average case hardness. Lecture Notes for Class (scribe: Elad Verbin) (2004), `http://www.cims.nyu.edu/ regev/teaching/lattices_fall_2004/ln/averagecase.pdf`
25. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, pp. 84–93 (2005)
26. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, 2014, May 31 - June 03, pp. 475–484. ACM (2014)
27. Schnorr, C.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. 53, 201–224 (1987)