Adaptively Secure Constrained Pseudorandom Functions in the Standard Model

Alex Davidson[1⋆], Shuichi Katsumata[2⋆⋆], Ryo Nishimaki[3], Shota Yamada[2⋆⋆], and Takashi Yamakawa[3]

[1] Cloudflare, Portugal
alex.davidson92@gmail.com
[2] AIST, Tokyo, Japan
{shuichi.katsumata,yamada-shota}@aist.go.jp
[3] NTT Secure Platform Laboratories, Tokyo, Japan
{ryo.nishimaki.zk, takashi.yamakawa.ga}@hco.ntt.co.jp

Abstract. Constrained pseudorandom functions (CPRFs) allow learning "constrained" PRF keys that can evaluate the PRF on a subset of the input space, or based on some predicate. First introduced by Boneh and Waters [AC'13], Kiayias et al. [CCS'13] and Boyle et al. [PKC'14], they have shown to be a useful cryptographic primitive with many applications. These applications often require CPRFs to be adaptively secure, which allows the adversary to learn PRF values and constrained keys in an arbitrary order. However, there is no known construction of adaptively secure CPRFs based on a standard assumption in the standard model for any non-trivial class of predicates. Moreover, even if we rely on strong tools such as indistinguishability obfuscation (IO), the state-of-the-art construction of adaptively secure CPRFs in the standard model only supports the limited class of $\mathbf{NC}^1$ predicates.

In this work, we develop new adaptively secure CPRFs for various predicates from different types of assumptions in the standard model. Our results are summarized below.

- We construct adaptively secure and $O(1)$-collusion-resistant CPRFs for t-conjunctive normal form (t-CNF) predicates from one-way functions (OWFs) where t is a constant. Here, $O(1)$-collusion-resistance means that we can allow the adversary to obtain a constant number of constrained keys. Note that t-CNF includes bit-fixing predicates as a special case.
- We construct adaptively secure and single-key CPRFs for inner-product predicates from the learning with errors (LWE) assumption. Here, single-key security means that we only allow the adversary to learn one constrained key. Note that inner-product predicates include t-CNF predicates for a constant t as a special case. Thus, this construction supports more expressive class of predicates than

⋆ Part of this work was completed while the author undertook a research internship at NTT when he was a PhD student at Royal Holloway. The author was also supported by the EPSRC and the UK Government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1).

⋆⋆ The authors were supported by JST CREST Grant Number JPMJCR19F6. The forth author was also supported by JSPS KAKENHI Grant Number 19H01109.

that supported by the first construction though it loses the collusion-resistance and relies on a stronger assumption.
- We construct adaptively secure and $O(1)$-collusion-resistant CPRFs for all circuits from the LWE assumption and indistinguishability obfuscation (IO).

The first and second constructions are the first CPRFs for any non-trivial predicates to achieve adaptive security outside of the random oracle model or relying on strong cryptographic assumptions. Moreover, the first construction is also the first to achieve any notion of collusion-resistance in this setting. Besides, we prove that the first and second constructions satisfy weak 1-key privacy, which roughly means that a constrained key does not reveal the corresponding constraint. The third construction is an improvement over previous adaptively secure CPRFs for less expressive predicates based on IO in the standard model.

1 Introduction

Pseudorandom functions (PRFs) provide the basis of a huge swathe of cryptography. Intuitively, such functions take a secret key and some binary string x as input, and output (deterministically) some value y. The pseudorandomness requirement dictates that y is indistinguishable from the output of a uniformly sampled function operating solely on x. PRFs provide useful sources of randomness in cryptographic constructions that take adversarially-chosen inputs. Many constructions of PRFs from standard assumptions are known, e.g., [25,38,37,7].

There have been numerous expansions of the definitional framework surrounding PRFs. In this work, we focus on a strand of PRFs that are known as *constrained* PRFs or CPRFs. CPRFs were first introduced by Boneh and Waters [15] alongside the concurrent works of Kiayias et al. [33] and Boyle et al. [17]. They differ from standard PRFs in that they allow users to learn *constrained* keys to evaluate the PRF only on a subset of the input space defined by a predicate. Let K be a master key used to compute the base PRF value and let K_C be the constrained key with respect to a predicate C. Then, the output computed using the master key $y = \mathsf{CPRF.Eval}(\mathsf{K}, x)$ can be evaluated using a constrained key K_C if the input x satisfies the constraint, i.e., $C(x) = 1$. However, if $C(x) = 0$, then the output y will remain pseudorandom from a holder of K_C. The expressiveness of a CPRF is based on the class of constraints $\mathcal{C}$ it supports, where the most expressive class is considered to be $\mathsf{P/poly}$.

Similarly to the security notion of standard PRFs, we require CPRFs to satisfy the notion of *pseudorandomness on constrained points*. Formally, the adversary is permitted to make queries for learning PRF evaluations on arbitrary points as with standard PRFs. The adversary is also permitted to learn constrained keys for any predicates $C_i \in \mathcal{C}$ where $i \in [Q]$ for $Q = \mathsf{poly}$.[1] The security requirement dictates that the CPRF remains pseudorandom on a target input x^* that has not been queried so far, where $C_i(x^*) = 0$ for all i. There have been several

[1] Throughout the introduction, poly will denote an arbitrary polynomial in the security parameter.

flavors of this security requirement that have been considered in previous works: when the adversary can query the constrained keys arbitrarily, then we say the CPRF is *adaptively* secure on constrained points; otherwise, if all the constrained keys must be queried at the outset of the game it is *selectively* secure.[2] When $Q > 1$, then we say the CPRF is Q-*collusion-resistant*. In case $Q = \mathsf{poly}$, we write poly-collusion-resistant and when $Q = 1$, we say it is a *single-key* CPRF. These two notions capture the requirements of CPRFs and satisfying both requirements (adaptively-secure, poly-collusion-resistant) is necessary for many applications of CPRFs [15]. For instance, in one of the most appealing applications of CPRFs such as length-optimal broadcast encryption schemes and non-interactive policy-based key exchanges, we require an adaptive and poly-collusion-resistant CPRF for an expressive class of predicates.

We focus on known constructions of CPRFs in the standard model from standard assumptions, that is, CPRFs that do not rely on the random oracle model (ROM) and non-standard assumptions such as indistinguishability obfuscation (IO) or multilinear maps. We notice that there exists no construction of adaptively secure CPRFs for any class of predicates in this standard setting. For instance, even if we consider the most basic puncturable or prefix-fixing predicates, we require the power of IO or the ROM to achieve adaptive security. (For an explanation on different types of predicates, we refer to the full version). Notably, even though we now have many CPRFs for various predicate classes from different types of standard assumptions such as the learning with errors (LWE) and Diffie-Hellman (DH) type assumptions [5,19,18,20,3,22,39], all constructions only achieve the weaker notion of selective security. In addition, other than selectively-secure CPRFs for the very restricted class of prefix-fixing predicates [5], all the above constructions of CPRFs provide no notion of Q-collusion-resistance for any $Q > 1$. Indeed, most constructions admit trivial collapses in security once more than one constrained key is exposed. A natural open question arises:

> *(Q1). Can we construct adaptively secure constrained PRFs for any class of predicates based on standard assumptions in the standard model; preferably with collusion-resistance?*

Next, we focus on CPRFs based on any models and assumptions. So far the best CPRF we can hope for — an optimal CPRF — (i.e., it supports the constraint class of $\mathsf{P/poly}$, it is adaptively secure, and it is poly-collusion resistant) is only known based on IO in the ROM [28]. The moment we restrict ourselves to the standard model without relying on the ROM, we can only achieve a weaker notion of CPRF regardless of still being able to use strong tools such as IO. Namely, the following three incomparable state-of-the-art CPRFs (based on IO in the standard model) do not instantiate one of the requirements of the optimal CPRF: [29] only supports the very limited class of puncturing predicates; [14] only achieves selective security; and [4] only achieves single-key security for the

[2] In general, we can upgrade selective security to adaptive security by complexity leveraging. However, we want to avoid this since complexity leveraging needs subexponentially hard assumptions.

limited class of $\mathbf{NC}^1$ predicates. Therefore, a second open question that we are interested in is:

> *(Q2). Can we construct an adaptively secure and Q-collusion resistant (for any $Q > 1$) constrained PRFs for the widest class of* $\mathsf{P/poly}$ *predicates in the standard model?*

Note that solving the above question for *all* $Q > 1$ will result in an optimal CPRF, which we currently only know how to construct in the ROM.

1.1 Our Contribution

In this study, we provide concrete solutions to the questions $(Q1)$ and $(Q2)$ posed above. We develop new *adaptively* secure CPRF constructions for various expressive predicates from a variety of assumptions *in the standard model*. We summarize our results below. The first two results are answers to $(Q1)$, and the last result is an answer to $(Q2)$.

1. We construct an adaptively secure and $O(1)$-collusion-resistant CPRF for t-conjunctive normal form (t-CNF) predicates from one-way functions (OWFs), where t is any constant. Here, $O(1)$-collusion-resistance means that it is secure against adversaries who learn a constant number of constrained keys. This is the first construction to satisfy *adaptive* security or *collusion-resistance* from any standard assumption and in the standard model regardless of the predicate class it supports. Our CPRF is based solely on the existence of OWFs. In particular, it is a much weaker assumption required than all other CPRF constructions for the bit-fixing predicate (which is a special case of t-CNF predicates) [15,14,20,3]. Previous works rely on either the LWE assumption, the decisional DH assumption, or multilinear maps.
2. We construct an adaptively secure and single-key CPRF for inner-product predicates from the LWE assumption. Although our second CPRF does not admit any collusion-resistance, inner-product predicates are a strictly wider class of predicates compared to the t-CNF predicates considered above. (See the full version.) All other lattice-based CPRFs supporting beyond inner-product predicates (NC^1 or $\mathsf{P/poly}$) [19,18,20,22,39] achieve only selective security and admits no collusion-resistance.
3. We construct an adaptively secure and $O(1)$-collusion-resistant CPRF for $\mathsf{P/poly}$ from IO and the LWE assumption. More specifically, we use IO and shift-hiding shiftable functions [39], where the latter can be instantiated from the LWE assumption. This is the first adaptively secure CPRF for the class of $\mathsf{P/poly}$ in the standard model (it further enjoys any notion of collusion-resistance). As stated above, current constructions of CPRFs in the standard model either: only support the limited class of puncturing predicates [29]; achieves only selective security [14]; or only achieves single-key security for the limited class of $\mathbf{NC}^1$ predicates [4].

We also note that our first two constructions satisfy (weak) 1-key privacy, previously coined by Boneh et al. [14] (see the full version for more details on the definition of key privacy).

Applications. As one interesting application, our CPRF for bit-fixing predicates can be used as a building block to realize adaptively-secure t-CNF attribute-based-encryption (ABE) based on lattices, as recently shown by Tsabary [41]. Other than identity-based encryption [1,21] and non-zero inner product encryption [32], this is the first lattice-based ABE satisfying adaptive-security for a non-trivial class of policies. The ABE scheme by Tsabary shows that other than their conventional use-cases, CPRFs may be a useful tool to achieve higher security of more advanced cryptographic primitives.

An attentive reader may wonder whether our CPRFs have any other applications. For instance, as Boneh and Waters proved [15], one can construct length-optimal broadcast encryption schemes from CPRFs for bit-fixing predicates. However, unfortunately, for these types of applications, we require Q-collusion-resistance where Q is an a-priori bounded polynomial. Therefore, we cannot plug in our construction for these types of applications. We leave it as an interesting open problem to progress our CPRF constructions to achieve Q-collusion-resistance for larger Q; achieving $Q = \omega(1)$ would already seem to require a new set of ideas.

Relation to the lower bound by Fuchsbauer et al. [24]. One may wonder how our adaptively secure CPRF relates to the lower bound of adaptively secure CPRFs proven by Fuchsbauer et al. [24]. They proved that we could not avoid exponential security loss to prove adaptive pseudorandomness of *the specific CPRF for bit-fixing predicates by Boneh and Waters based on multilinear maps* [15]. Fortunately, their proofs rely heavily on the checkability of valid constrained keys by using multilinear maps. Therefore, their lower bounds do not apply to our setting since none of our constructions have checkability.

Comparison with Existing Constructions. There are several dimensions to consider when we compare CPRF constructions. In this section, we focus on *adaptively* secure CPRFs as it is one of our main contributions. Along with related works, a more extensive comparison is provided in the full version. The following Table 1 lists all the adaptively secure CPRFs known thus far. One clear advantage of our first two CPRFs is that they are the first CPRF to achieve adaptive security without relying on IO or the ROM. However, it can be seen that this comes at the cost of supporting a weaker predicate class, or achieving single-key or $O(1)$-collusion-resistance. Regarding our third CPRF, the main advantage is that it achieves adaptive security and supports the broadest predicate class $\mathsf{P/poly}$ without resorting to the ROM. Compared to the recent CPRF by Attrapadung et al. [4], we provide a strict improvement since our first construction supports $O(1)$-collusion-resistance.

Historical Note on Our First Contribution. In the initial version of this paper, we gave a construction of adaptively secure and $O(1)$-collusion-resistant CPRFs for bit-fixing predicates. After the initial version, Tsabary [41] observed that essentially the same idea could be used to construct adaptively single-key secure CPRFs for t-CNF predicates for a constant t. We further extend her construction

Table 1. Comparison among adaptively secure CPRFs. In column "Predicate", LR, BF, t-CNF, and IP stand for left-right-fixing, bit-fixing, t-conjunctive normal form, and inner-product predicates, respectively. In column "Assumption", BDDH, LWE, SGH, and L-DDHI stand for bilinear decisional Diffie-Hellman, multilinear decisional Diffie-Hellman, learning with errors, subgroup hiding assumption, and L-decisional Diffie-Hellman inversion assumptions, respectively. Regarding key privacy, ‡ means that this satisfies *weak* key privacy.

	Adaptive	Collusion-resistance	Privacy	Predicate	Assumption
BW [15]	✓	poly	poly	LR	BDDH & ROM
HKKW [28]	✓	poly	0	P/poly	IO & ROM
HKW [29]	✓	poly	0	Puncturing	SGH & IO
AMNYY [3]	✓	1	1	BF	ROM
	✓	1	0	NC^1	L-DDHI & ROM
AMNYY [4]	✓	1	0	NC^1	SGH & IO
Sec. 4	✓	$O(1)$	$1^{\ddagger}$	t-CNF ($\supseteq$ BF)	OWF
Sec. 5	✓	1	$1^{\ddagger}$	IP	LWE
Sec. 6	✓	$O(1)$	0	P/poly	LWE & IO

to construct adaptively secure and $O(1)$-collusion-resistant CPRFs for t-CNF predicates for a constant t in the current version. We stress that (the initial version of) this paper is the first to give adaptively secure or collusion-resistant CPRFs under a standard assumption and in the standard model, for any non-trivial class of predicates.

2 Technical Overview

In this section, we explain the approach we took for achieving each of our CPRFs. For CPRFs for bit-fixing (and t-CNF) predicates, we take a combinatorial approach. For CPRFs for inner-product predicates, we take an algebraic approach based on lattices incorporating the so-called lossy mode. For CPRFs for P/poly, we use shift-hiding shiftable functions [39] and IO as main building blocks. In the subsequent subsections, we explain these approaches in more detail.

2.1 CPRF for Bit-Fixing/t-CNF

We achieve CPRFs for t-CNF predicates. However, we consider our CPRF for bit-fixing predicates in the technical overview rather than the more general CPRF for t-CNF predicates for ease of presentation. The high-level idea is very similar and generalizes naturally. Here, a bit-fixing predicate is defined by a string $v \in \{0,1,*\}^{\ell}$ where $*$ is called the "wildcard". A bit-fixing predicate v on input $x \in \{0,1\}$ is said to be satisfied if and only if $(v_i = x_i) \vee (v_i = *)$ for all $i \in [\ell]$.

We first focus on how to achieve collusion-resistance because the structure for achieving collusion-resistance naturally induces adaptive security.

Combinatorial Techniques for CPRFs for bit-fixing predicates. We start with a simpler case of *single-key* CPRF for bit-fixing predicates as our starting point. We use 2ℓ keys of standard PRFs to construct an ℓ-bit input CPRF for bit-fixing predicates. Let $\mathsf{PRF.Eval} : \{0,1\}^\kappa \times \{0,1\}^\ell \mapsto \{0,1\}^n$ be the evaluation algorithm of a PRF. We uniformly sample keys $\mathsf{K}_{i,b} \in \{0,1\}^\kappa$ for $i \in [\ell]$ and $b \in \{0,1\}$. The master key of the CPRF is $\mathsf{K} = \{\mathsf{K}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ and evaluation on some $x \in \{0,1\}^\ell$ is computed as the output of:

$$\mathsf{CPRF.Eval}(\mathsf{K}, x) = \bigoplus_{i=1}^{\ell} \mathsf{PRF.Eval}(\mathsf{K}_{i,x_i}, x).$$

Figure 1 depicts the construction.

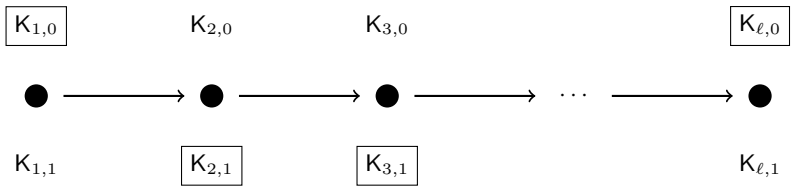

Fig. 1. Length-ℓ directed line representation where each nodes are labeled with two PRF keys. In the figure, the choices of PRF keys correspond to some input $x = 011\cdots0$.

The constrained key for a bit-fixing predicate $v \in \{0,1,*\}^\ell$ constitutes a single PRF key K_{i,v_i}(where $v_i \in \{0,1\}$), and a pair of PRF keys $(\mathsf{K}_{i,0}, \mathsf{K}_{i,1})$ (where $v_i = *$). Constrained evaluation is clearly possible for any input x that satisfies the bit-fixing predicate v since we have keys K_{i,v_i} for non-wildcard parts and both keys $(\mathsf{K}_{i,0}, \mathsf{K}_{i,1})$ for wildcard parts.

The (selective) security of the scheme rests upon the fact that for a single constrained key, with respect to v, there must exist a $j \in [\ell]$ such that $(x^*_j \neq v_j) \wedge (v_j \neq *)$ for the challenge input x^*. This is due to the fact that the bit-fixing predicate v does not satisfy x^*. Then, pseudorandomness of $y \leftarrow \mathsf{CPRF.Eval}(\mathsf{K}, x^*)$ is achieved because

$$y \leftarrow \bigoplus_{i=1}^{\ell} \mathsf{PRF.Eval}(\mathsf{K}_{i,x^*_i}, x^*) = \mathsf{PRF.Eval}(\mathsf{K}_{j,x^*_j}, x^*) \oplus \left(\bigoplus_{i\neq j} \mathsf{PRF.Eval}(\mathsf{K}_{i,x^*_i}, x^*) \right)$$

where $\mathsf{PRF.Eval}(\mathsf{K}_{j,x^*_j}, x^*)$ is evaluated using the key that is unknown to the adversary. Thus, this evaluation can be replaced with a uniformly sampled $y_j \in \{0,1\}^n$ by the pseudorandomness of PRF for key K_{j,x^*_j}. In turn, this results in a uniformly distributed CPRF output y and so pseudorandomness is ensured. We can instantiate pseudorandom functions using only one-way functions [25,27], and therefore, so can the above single-key CPRF for the bit-fixing predicate.

Allowing > 1 *constrained key query.* If we allow for more than two constrained key queries in the above construction, the scheme is trivially broken. Consider an adversary that queries the two bit-fixing predicates $v = 0 * * \ldots * *0$ and $\bar{v} = 1 * * \ldots * *1$ as an example. Notice that any binary string x of the form $x = 0 \ldots 1$ or $x = 1 \ldots 0$ will not satisfy either of the predicates. Therefore, we would like the evaluation value y on such input x by the master key to remain pseudorandom to the adversary. However, the adversary will be able to collect all PRF keys $\{\mathsf{K}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ by querying v and $\bar{v}$, and recover the master key itself in our construction above. Therefore, the adversary will be able to compute on any input x regardless of its constraints.

Collusion-resistance for two constrained key queries. At a high level, the reason why our construction could not permit more than one constrained key query is because we examined each of the input bits individually when choosing the underlying PRF keys. Now, consider a scheme that considered two input bits instead of considering one input bit at each node in Figure 1. Figure 2 illustrates this modified construction. In the set-up shown in Figure 2 at each node (i, j), we now consider the i^{th} and j^{th} input bits of the string $x \in \{0,1\}^\ell$ and choose the key $\mathsf{K}_{(i,j),(b_1,b_2)}$ where $b_1 = x_i$ and $b_2 = x_j$; the master key is the combination of all such keys $\mathsf{K} = \{\mathsf{K}_{(i,j),(b_1,b_2)}\}_{(i,j)\in[\ell]^2,(b_1,b_2)\in\{0,1\}^2}$.

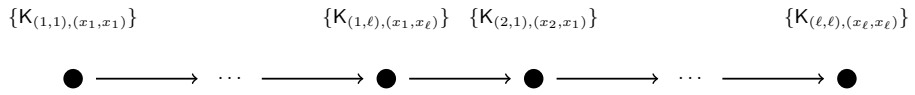

Fig. 2. Length-ℓ^2 directed line representation where each nodes consider two input bits, where $(x_i, x_j) \in \{0,1\} \times \{0,1\}$ for all $i, j \in [\ell]$.

Evaluation is then carried out by adding the PRF values along the directed line illustrated in Figure 2:

$$\mathsf{CPRF.Eval}(\mathsf{K}, x) = \bigoplus_{(i,j)\in[\ell]\times[\ell]} \mathsf{PRF.Eval}(\mathsf{K}_{(i,j),(x_i,x_j)}, x),$$

and constrained keys for $v \in \{0,1,*\}^\ell$ contain the key $\mathsf{K}_{(i,j),(b_1,b_2)}$, for all $b_1, b_2 \in \{0,1\}$ such that

$$\Big((v_i = b_1) \vee (v_i = *)\Big) \bigwedge \Big((v_j = b_2) \vee (v_j = *)\Big),$$

is satisfied.

To see how this combinatorial change in the construction has an impact on the collusion-resistance of the scheme, consider a pair of constrained key queries for bit-fixing predicates $v, \bar{v} \in \{0,1,*\}^\ell$. Let x^* be the challenge input that is constrained with respect to both $v, \bar{v}$. Then there exists an $i' \in [\ell]$ where $(x^*_{i'} \neq v_{i'}) \wedge (v_{i'} \neq *)$ and likewise $(x^*_{j'} \neq \bar{v}_{j'}) \wedge (\bar{v}_{j'} \neq *)$ for some $j' \in [\ell]$.

Equivalently, we must have $x^*_{i'} = 1 - v_{i'}$ and $x^*_{j'} = 1 - \bar{v}_{j'}$ for some $i', j' \in [\ell]$. As a result, for these constrained key queries we observe that the underlying PRF key $\mathsf{K}_{(i',j'),(1-v_{i'},1-\bar{v}_{j'})}$ will never be revealed to the adversary.

Using this fact, we can prove that our new CPRF construction achieves collusion-resistance for two constrained key queries using essentially the same aforementioned proof technique. We rewrite the CPRF evaluation on x^* as:

$$\begin{aligned}\mathsf{CPRF.Eval}(\mathsf{K}, x^*) &= \bigoplus_{(i,j)\in[\ell]\times[\ell]} \mathsf{PRF.Eval}(\mathsf{K}_{(i,j),(x^*_i,x^*_j)}, x^*) \\ &= \mathsf{PRF.Eval}(\mathsf{K}_{(i',j'),(x^*_{i'},x^*_{j'})}, x^*) \oplus \left(\bigoplus_{(i,j)\neq(i',j')} \mathsf{PRF.Eval}(\mathsf{K}_{(i,j),(x^*_i,x^*_j)}, x^*) \right).\end{aligned}$$

Notice that, since $\mathsf{K}_{(i',j'),(x^*_{i'},x^*_{j'})}$ is never revealed to the adversary, this evaluation is indistinguishable from a uniformly sampled value y^*. In a simulation where y^* replaces the underlying PRF evaluation, the entire CPRF evaluation on x^* is distributed uniformly and pseudorandomness follows accordingly.

Expanding to $O(1)$-collusion-resistance. The technique that we demonstrate in this work is a generalisation of the technique that we used for two-key collusion-resistance. Instead of considering two input bits at a time, we consider Q input bits at a time and index each node in the evaluation by the vector $(i_1, \ldots, i_Q) \in [\ell]^Q$. Then we evaluate the CPRF on $x \in \{0,1\}^\ell$ as the output of:

$$\mathsf{CPRF.Eval}(\mathsf{K}, x) = \bigoplus_{(i_1,\ldots,i_Q)\in[\ell]^Q} \mathsf{PRF.Eval}(\mathsf{K}_{(i_1,\ldots,i_Q),(x_{i_1},\ldots,x_{i_Q})}, x).$$

The constraining algorithm works for a bit-fixing predicate defined by $v \in \{0,1,*\}^\ell$ by providing all keys $\mathsf{K}_{(i_1,\ldots,i_Q),(b_1,\ldots,b_Q)}$ such that

$$\bigwedge_{j\in[Q]} (b_j = v_{i_j}) \vee (v_{i_j} = *)$$

is satisfied. Constrained evaluation is then possible for any input x satisfying the bit-fixing predicate defined by v.

For any set of Q constrained key queries associated with strings $v^{(1)}, \ldots, v^{(Q)}$ and any constrained input x^*, there must exist a vector $(i'_1, \ldots, i'_Q)$ such that $(x^*_{i'_j} \neq v^{(j)}_{i'_j}) \wedge (v^{(j)}_{i'_j} \neq *)$ for all $j \in [Q]$. Therefore, the key $\mathsf{K}_{(i'_1,\ldots,i'_Q),(x^*_{i'_1},\ldots,x^*_{i'_Q})}$ is never revealed to the adversary. Finally, we can prove the selective pseudorandomness of the CPRF on input x^* using exactly the same technique as mentioned in the case when $Q = 2$. The proof of security is given in the proof of Theorem 4.1.

Importantly, we cannot achieve collusion-resistance for unbounded Q because there is an exponential dependency on Q associated with the size of the CPRF. For instance, for the node indexed by the vector $(i_1, \ldots, i_Q)$, there are 2^Q underlying PRF keys associated with this node; moreover, there are ℓ^Q such nodes. Therefore

the total size of K is $(2\ell)^Q$. As a result, we are only able to afford $Q = O(1)$ since ℓ is the input length of PRF, which is a polynomial in the security parameter. This bound is inherent in the directed line paradigm because our technique is purely combinatorial.

Finally, we assess the security properties achieved by our CPRF for bit-fixing predicates. Although we have been showing selective security of our CPRF, we observe that our construction satisfies adaptive security when the underlying pseudorandom functions satisfy adaptive pseudorandomness.

Achieving Adaptive security. Our construction arrives at adaptive security essentially *for free*. Previous constructions for bit-fixing predicates (or as a matter of fact, any non-trivial predicates) incur sub-exponential security loss during the reduction from adaptive to selective security, or relies on the random oracle model or IO; see the full version for an overview. The sub-exponential security loss is incurred as previous constructions achieve adaptive security by letting the reduction guess the challenge input x^* that the adversary chooses.

We can achieve adaptive security with a polynomial security loss (e.g. $1/\mathsf{poly}(\kappa)$): by instead guessing the key (not the challenge input) that is implicitly used by the adversary (i.e. K_{T^*,x^*_T} for $T^* \subset [\ell], |T^*| = Q$). For example in the 2-key setting explained above, this amounts to correctly guessing the values (i, j) and (x^*_i, x^*_j) of the PRF key $\mathsf{K}_{(i,j),(x^*_i,x^*_j)}$, which happens with probability at most $(1/2\ell)^2$. If this key is not eventually used by the challenge ciphertext, or it is revealed via a constrained key query, then the reduction algorithm aborts. This is because *the entire proof hinges on the choice of this key*, rather than the input itself. Since there are only polynomially many keys (for $Q = O(1)$), we can achieve adaptive security with only a $1/\mathsf{poly}(\kappa)$ probability of aborting.

Finally, we note that there is a subtle technical issue we must resolve due to the non-trivial abort condition. Similar problems were identified by Waters [42] who introduced the "artificial abort step".

2.2 CPRF for Inner-Product

We construct CPRF for the class of inner-product predicates (over the integers) based on lattices.

The starting point of our CPRF is the lattice-based PRF of [13,6]. At a very high level, the secret key K of these PRFs is a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and the public parameters is some matrices $(\mathbf{A}_i \in \mathbb{Z}_q^{n\times m})_{i\in[k]}$. To evaluate on an input $\mathbf{x}$, one first generates a (publicly computable) matrix $\mathbf{A}_\mathbf{x} \in \mathbb{Z}_q^{n\times m}$ related to input $\mathbf{x}$ and simply outputs the value $\lfloor \mathbf{s}^\top \mathbf{A}_\mathbf{x} \rfloor_p \in \mathbb{Z}_p^m$, where $\lfloor a \rfloor_p$ denotes rounding of an element $a \in \mathbb{Z}_q$ to $\mathbb{Z}_p$ by multiplying it by (p/q) and rounding the result. Roughly, the values $\mathbf{s}^\top \mathbf{A}_\mathbf{x} + \mathsf{noise}$ are jointly indistinguishable from uniform for different inputs $\mathbf{x}$ since $\mathbf{A}_\mathbf{x}$ acts as an LWE matrix. Therefore, if the noise term is sufficiently small, then $\lfloor \mathbf{s}^\top \mathbf{A}_\mathbf{x} \rfloor_p = \lfloor \mathbf{s}^\top \mathbf{A}_\mathbf{x} + \mathsf{noise} \rfloor_p$, and hence, pseudorandomness follows.

Pioneered by the lattice-based CPRF of Brakerski and Vaikuntanathan [19], many constructions of CPRF [12,18] have built on top of the PRF of [13,6]. The

high-level methodology is as follows: the constrained key for a constraint C would be a set of LWE ciphertexts of the form $\mathsf{K}_C := (\mathsf{ct}_i = \mathbf{s}^\top(\mathbf{A}_i - C_i \cdot \mathbf{G}) + \mathsf{noise})_{i\in[k]}$, where C_i is the i^{th} bit of the description of the constraint C and $\mathbf{G}$ is the so-called gadget matrix [36]. To evaluate on input $\mathbf{x}$ using the constrained key K_C, one evaluates the ciphertexts $(\mathsf{ct}_i)_{i\in[k]}$ to $\mathsf{ct}_{\mathbf{x}} = \mathbf{s}^\top(\mathbf{A}_{\mathbf{x}} - (1 - C(\mathbf{x})) \cdot \mathbf{G}) + \mathsf{noise}$, using the by now standard homomorphic computation technique of [11] originally developed for attribute-based encryption (ABE) schemes. Here, $\mathbf{A}_{\mathbf{x}}$ is independent of the constraint C, that is, $\mathbf{A}_{\mathbf{x}}$ can be computed without the knowledge of C. Then, the final output of the CPRF evaluation with the constrained key will be $\lfloor \mathsf{ct}_{\mathbf{x}} \rfloor_p$. Now, if the constraint is satisfied, i.e., $C(\mathbf{x}) = 1$, then computing with the constrained key K_C will result in the same output as the master key K since we would have $\mathsf{ct}_{\mathbf{x}} = \mathbf{s}^\top \mathbf{A}_{\mathbf{x}} + \mathsf{noise}$.

Unfortunately, all works which follow this general methodology only achieves *selective* security. There is a noted resemblance between this construction with the above types of CPRF and the ABE scheme of [11]. As a consequence, achieving an adaptively secure CPRF following the above methodology would likely shed some light onto the construction of an adaptively secure lattice-based ABE. Considering that adaptively secure ABEs are known to be one of the major open problems in lattice-based cryptography, it does not seem to be an easy task to achieve an adaptively secure CPRF following this approach.

We take a different approach by taking advantage of the fact that our constraint is a simple linear function in this work due to the technical hurdle above. Specifically, we only embed the constraint in the master key $\mathbf{s}$ instead of embedding the constraint in the master key $\mathbf{s}$ *and* the public matrices $(\mathbf{A}_i)_{i\in[k]}$ as $(\mathbf{s}^\top(\mathbf{A}_i - C_i \cdot \mathbf{G}))_{i\in[k]}$. To explain this idea, we need some preparation. Let $\mathbf{y} \in \mathbb{Z}^\ell$ be the vector associated with the inner-product constraint $C_{\mathbf{y}}$, that is, the constrained key $\mathsf{K}_{C_{\mathbf{y}}}$ can evaluate on input $\mathbf{x} \in \mathbb{Z}^\ell$ if and only if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ (over the integers). We also slightly modify the PRF of [13,6] so that we use a matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times\ell}$ instead of a vector $\mathbf{s} \in \mathbb{Z}_q^n$ as the secret key. To evaluate on input $\mathbf{x} \in \mathbb{Z}^\ell$ with the secret key $\mathbf{S}$, we will first compute the vector $\mathbf{s}_{\mathbf{x}} = \mathbf{S}\mathbf{x} \in \mathbb{Z}^n$ and then run the PRF of [13,6], viewing $\mathbf{s}_{\mathbf{x}}$ as the secret key. That is, the output of the PRF is now $\lfloor \mathbf{s}_{\mathbf{x}}^\top \mathbf{A}_{\mathbf{x}} \rfloor_p$.

The construction of our CPRF is a slight extension of this. The master key and evaluation with the master key is the same as the modified PRF. Namely, the master key is defined as $\mathsf{K} := \mathbf{S}$ and the output of the evaluation is $\lfloor \mathbf{s}_{\mathbf{x}}^\top \mathbf{A}_{\mathbf{x}} \rfloor_p$. Our constrained key for the constraint $C_{\mathbf{y}}$ is then defined as $\mathsf{K}_{C_{\mathbf{y}}} := \mathbf{S}_{\mathbf{y}} = \mathbf{S} + \mathbf{d} \otimes \mathbf{y}^\top \in \mathbb{Z}_q^{n\times\ell}$ where $\mathbf{d}$ is a uniformly random vector sampled over $\mathbb{Z}_q^n$. Evaluation with the constrained key $\mathsf{K}_{C_{\mathbf{y}}} = \mathbf{S}_{\mathbf{y}}$ is done exactly the same as with the master key $\mathsf{K} = \mathbf{S}$; it first computes $\mathbf{s}_{\mathbf{y},\mathbf{x}} = \mathbf{S}_{\mathbf{y}}\mathbf{x}$ and outputs $\lfloor \mathbf{s}_{\mathbf{y},\mathbf{x}}^\top \mathbf{A}_{\mathbf{x}} \rfloor_p$. It is easy to check that if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ (i.e., $C_{\mathbf{y}}(\mathbf{x}) = 1$), then $\mathbf{S}_{\mathbf{y}}\mathbf{x} = (\mathbf{S} + \mathbf{d} \otimes \mathbf{y}^\top)\mathbf{x} = \mathbf{s}_{\mathbf{x}}$. Hence, the constrained key computes the same output as the master key for the inputs for which the constraint is satisfied. The construction is very simple, but the proof for adaptive security requires a bit of work.

As a warm-up, let us consider the easy case of selective security and see why it does not generalize to adaptive security. When the adversary $\mathcal{A}$ submits $C_{\mathbf{y}}$

as the challenge constraint at the beginning of the selective security game, the simulator samples $\hat{\mathbf{S}} \xleftarrow{\$} \mathbb{Z}_q^{n \times \ell}$ and $\mathbf{d} \xleftarrow{\$} \mathbb{Z}_q^n$; sets the master key as $\mathsf{K} = \hat{\mathbf{S}} - \mathbf{d} \otimes \mathbf{y}^\top$ and the constrained key as $\mathsf{K}_{C_\mathbf{y}} = \hat{\mathbf{S}}$; and returns $\mathsf{K}_{C_\mathbf{y}}$ to $\mathcal{A}$. Since the distribution of K and $\mathsf{K}_{C_\mathbf{y}}$ is exactly the same as in the real world, the simulator perfectly simulates the keys to $\mathcal{A}$. Now, notice that evaluation on input $\mathbf{x}$ with the master key K results as

$$\begin{aligned}\mathbf{z} &= \left\lfloor \left((\hat{\mathbf{S}} - \mathbf{d} \otimes \mathbf{y}^\top)\mathbf{x} \right)^\top \mathbf{A}_\mathbf{x} \right\rfloor_p \\ &\approx \lfloor (\hat{\mathbf{S}}\mathbf{x})^\top \mathbf{A}_\mathbf{x} \rfloor_p - \langle \mathbf{x}, \mathbf{y} \rangle \cdot \lfloor \mathbf{d}^\top \mathbf{A}_\mathbf{x} \rfloor_p = \mathsf{CPRF}_{\mathsf{K}_{C_\mathbf{y}}}(\mathbf{x}) - \langle \mathbf{x}, \mathbf{y} \rangle \cdot \mathsf{PRF}_\mathbf{d}(\mathbf{x}),\end{aligned}$$

where $\mathsf{CPRF}_{\mathsf{K}_{C_\mathbf{y}}}(\mathbf{x})$ is the CPRF evaluation with constrained key $\mathsf{K}_{C_\mathbf{y}}$ and $\mathsf{PRF}_\mathbf{d}(\mathbf{x})$ is the PRF evaluation of [13,6] with secret key $\mathbf{d} \in \mathbb{Z}_q^n$.[3] In particular, the simulator can simply reply to the evaluation query $\mathbf{x}$ made by $\mathcal{A}$ by first evaluating $\mathbf{x}$ with the constrained key $\mathsf{K}_{C_\mathbf{y}}$ and then shifting it by $\langle \mathbf{x}, \mathbf{y} \rangle \cdot \mathsf{PRF}_\mathbf{d}(\mathbf{x})$. With this observation, selective security readily follows from the security of the underlying PRF. Specifically, $\mathcal{A}$ will obtain many output values $\mathsf{PRF}_\mathbf{d}(\mathbf{x})$ for any $\mathbf{x}$ of its choice in the course of receiving $\mathbf{z}$ back on an evaluation query on input $\mathbf{x}$. However, $\mathsf{PRF}_\mathbf{d}(\mathbf{x}^*)$ will remain pseudorandom for a non-queried input $\mathbf{x}^*$ due to the security of the PRF. Hence, the challenge output $\mathbf{z}^*$ will remain pseudorandom from the view of $\mathcal{A}$.

Unfortunately, the above approach breaks down if we want to show adaptive security. This is because the simulator will no longer be able to simulate the "shift" $\langle \mathbf{x}, \mathbf{y} \rangle \cdot \mathsf{PRF}_\mathbf{d}(\mathbf{x})$ if it does not know the vector $\mathbf{y}$ associated with the challenge constraint $C_\mathbf{y}$. In particular, it seems the simulator is bound to honestly compute the master key $\mathsf{K} = \mathbf{S}$ and to use K to answer the evaluation query made before the challenge constraint query. Therefore, to cope with this apparent issue, we deviate from the above approach used to show selective security.

Our high-level approach for adaptive security will be to argue that $\mathbf{d}$ retains sufficient min-entropy conditioned on the view of $\mathcal{A}$, where $\mathcal{A}$ obtains a constrained key $\mathsf{K}_{C_\mathbf{y}} = \mathbf{S}_\mathbf{y}$ and honest evaluation on inputs $(\mathbf{x}_j)_{j \in [Q]}$ where Q is an arbitrary polynomial. Intuitively, if $\mathbf{d} \in \mathbb{Z}_q^n$ retains enough min-entropy, then it will mask part of the master key $\mathbf{S}$ conditioned on $\mathcal{A}$'s knowledge on $\mathbf{S}_\mathbf{y} = \mathbf{S} + \mathbf{d} \otimes \mathbf{y}^\top$, and hence, we would be able to argue that the output evaluated using the master key $\mathbf{S}$ is pseudorandom using some randomness extractor-type argument.

The proof for adaptive security is roughly as follows: Let $\mathsf{K} = \mathbf{S}$. The simulator will basically run identically to the challenger in the real world. It will honestly answer to $\mathcal{A}$'s evaluation query on input $\mathbf{x}$ by returning $\lfloor (\mathbf{S}\mathbf{x})^\top \mathbf{A}_\mathbf{x} \rfloor_p$ computed via the master key. When $\mathcal{A}$ queries for a constrained key on constraint $C_\mathbf{y}$, the simulator honestly responds by returning $\mathsf{K}_{C_\mathbf{y}} = \mathbf{S}_\mathbf{y}$. Evaluation queries after the constrained key query will also be answered using the master key. Then, similarly to the above equation, the output $\mathbf{z}$ returned to $\mathcal{A}$ as an evaluation query on

[3] Note that *a lot* of subtlety on parameter selections and technicalities regarding rounding are swept under the rug. However, we believe the rough details are enough to convey the intuition.

input $\mathbf{x}$ can be written as

$$\begin{aligned}\mathbf{z} &= \lfloor (\mathbf{S}\mathbf{x})^\top \mathbf{A}_\mathbf{x} \rfloor_p \\ &\approx \lfloor (\mathbf{S}_\mathbf{y}\mathbf{x})^\top \mathbf{A}_\mathbf{x} \rfloor_p - \langle \mathbf{x}, \mathbf{y} \rangle \cdot \lfloor \mathbf{d}^\top \mathbf{A}_\mathbf{x} \rfloor_p = \mathsf{CPRF}_{\mathsf{K}_{C_\mathbf{y}}}(\mathbf{x}) - \langle \mathbf{x}, \mathbf{y} \rangle \cdot \lfloor \mathbf{d}^\top \mathbf{A}_\mathbf{x} \rfloor_p.\end{aligned}$$

Therefore, conditioned on $\mathcal{A}$'s view, each query will leak information of $\mathbf{d}$ through the term $\lfloor \mathbf{d}^\top \mathbf{A}_\mathbf{x} \rfloor_p$. Moreover, if we run the standard homomorphic computation of [11], $\mathbf{A}_\mathbf{x}$ will be a full-rank matrix with overwhelming probability, and hence, $\lfloor \mathbf{d}^\top \mathbf{A}_\mathbf{x} \rfloor_p$ may uniquely define $\mathbf{d}$. Notably, information theoretically, everything about $\mathbf{d}$ may completely leak through a *single* evaluation query. Therefore, the question to be solved is: how can we restrict the information of $\mathbf{d}$ leaked through the evaluation query?

The main idea to overcome this problem is to use the *lossy mode* of the LWE problem [26,8,2,34]. The lossy LWE mode is a very powerful tool which states that if we sample $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ from a special distribution which is computationally indistinguishable from random (assuming the hardness of LWE), then $(\mathbf{A}, \mathbf{d}^\top \mathbf{A} + \mathsf{noise})$ leaks almost no information on $\mathbf{d}$. We call such a matrix $\mathbf{A}$ as "lossy". Our idea draws inspiration from the recent work of Libert, Stehlé, and Titiu [35] that shows that this lossy LWE mode can be combined with homomorphic computation of [11] to obtain adaptively secure distributed lattice-based PRFs. We will setup the public matrices $(\mathbf{A}_i)_{i \in [k]}$ in a special way during the simulation. Concretely, the special setup induces a lossy matrix on all the evaluation queries and a non-lossy matrix (i.e., $(\mathbf{A}_\mathbf{x}, \mathbf{d}^\top \mathbf{A}_\mathbf{x} + \mathsf{noise})$ uniquely defines $\mathbf{d}$) on the challenge query with non-negligible probability when we homomorphically compute $\mathbf{A}_\mathbf{x}$. For the knowledgeable readers, this programming of $\mathbf{A}_\mathbf{x}$ is accomplished by using admissible hash functions [10]. With this idea in hand, we will be able to argue that each evaluation query will always leak the same information on $\mathbf{d}$. Then, we will be able to argue that $\mathbf{z}^* = \lfloor \mathbf{d}^\top \mathbf{A}^*_\mathbf{x} \rfloor_p$ will have high min-entropy conditioned on $\mathcal{A}$'s view since $\mathbf{A}^*_\mathbf{x}$ will be a non-lossy matrix on the challenge input $\mathbf{x}^*$. Finally, we will use a deterministic randomness extractor to extract statistically uniform bits from $\mathbf{z}^*$.

We end this part by noting that $\mathsf{K} = \mathbf{S}$ and $\mathbf{d}$ will be taken from a more specific domain and there will be many subtle technical issues regarding the rounding operation in our actual construction. Moreover, similarly to [35], there are subtle issues on why we have to resort to *deterministic* randomness extractors and not any randomness extractors. For more detail, see Sec. 5.

2.3 CPRF for P/poly

Our CPRF for $\mathsf{P/poly}$ is constructed based on IO and shift-hiding shiftable functions (SHSF) [39].

First, we briefly recall SHSF. An SHSF consists of the following algorithms: a key generation algorithm $\mathsf{SHSF.KeyGen}$, which generates a master key msk; an evaluation algorithm $\mathsf{SHSF.Eval}$, which takes msk and $x \in \mathcal{X}$ as input and outputs $y \in \mathcal{Y}$; a shifting algorithm $\mathsf{SHSF.Shift}$, which takes msk and a function $C : \mathcal{X} \to \mathcal{Y}$ as input and outputs a shifted secret key sk_C; and a shifted evaluation

algorithm $\mathsf{SHSF.SEval}$, which takes a shifted evaluation key sk_C and $x \in \mathcal{X}$ as input and outputs $y \in \mathcal{Y}$. As correctness, we require that $\mathsf{SHSF.SEval}(\mathsf{sk}_C, x) \approx \mathsf{SHSF.Eval}(\mathsf{msk}, x) + C(x)$ holds where $+$ denotes an appropriately defined addition in $\mathcal{Y}$ and $\approx$ hides a small error. In this overview, we neglect the error and assume that this equation exactly holds for simplicity. The security of SHSF roughly says that sk_C does not reveal the shifting function C. More precisely, we require that there exists a simulator $\mathsf{SHSF.Sim}$ that simulates sk_C without knowing C so that it is computationally indistinguishable from an honestly generated one.

Before going into detail on our CPRF, we make one observation, which simplifies our security proof. Specifically, we can assume that an adversary does not make an evaluation query without loss of generality when we consider a (constant) collusion-resistant CPRF for $\mathsf{P/poly}$. This is because we can replace polynomial number of evaluation queries with one extra constrained key query on a "partitioning function" by the standard partitioning technique. (See Lemma 6.2 and its proof in the full version for the detail.) Thus, we assume that an adversary does not make any evaluation query at all, and only makes constrained key queries and a challenge query in the following.

We describe our construction of CPRF. A master key K of the CPRF is a secret key $\mathsf{sk}^{\mathsf{sim}}$ of SHSF generated by $\mathsf{SHSF.Sim}$, and the evaluation algorithm of the CPRF with the master key $\mathsf{K} = \mathsf{sk}^{\mathsf{sim}}$ is just defined as $\mathsf{SHSF.SEval}(\mathsf{sk}^{\mathsf{sim}}, \cdot)$. A constrained key K_C for a circuit C is defined to be an obfuscated program in which $\mathsf{sk}^{\mathsf{sim}}$ and C are hardwired and that computes $\mathsf{SHSF.SEval}(\mathsf{sk}^{\mathsf{sim}}, x)$ if $C(x) = 1$ and returns $\bot$ otherwise. This construction clearly satisfies the correctness of CPRF.

In the following, we show that this CPRF is adaptively secure against adversaries that make $O(1)$ constrained key queries and no evaluation query, which is sufficient to obtain $O(1)$ collusion-resistant adaptive CPRF that tolerates polynomial number of evaluation queries as explained above. First, we remark that constrained key queries made after the challenge query are easy to deal with. Namely, we can replace the master key hardwired into the constrained keys with a "punctured key" that can evaluate the CPRF on all inputs except for the challenge input by using the security of IO and the shift-hiding property of SHSF. Then, we can argue that the challenge output is still pseudorandom even given these constrained keys. We omit the details since this is a simple adaptation of the standard puncturing technique [40,16]. In the following, we assume that all constrained key queries are made before the challenge query so that we can focus on the most non-trivial part.

We begin by considering the single-key security, and later explain how to extend the proof to the $O(1)$-collusion-resistant case. In the single-key security game, an adversary only makes one constrained key query C and a challenge query x^* in this order. Recall that we are assuming that an adversary does not make any evaluation query and does not make any constrained key query after a challenge query is made without loss of generality. The main observation is that the simulator can generate the master key K *with knowledge of the constraint* C associated to the constrained key query since it can postpone generation of K

until a constrained key query is made. For proving the security in this setting, we consider the following game hops.

In the first, we replace the master key $\mathsf{K} = \mathsf{sk}^{\mathsf{sim}}$ with a shifted secret key sk_1 generated by $\mathsf{SHSF.Shift}(\mathsf{msk}_1, \overline{C}(\cdot) \cdot r)$. Here, $\mathsf{msk}_1 \xleftarrow{\$} \mathsf{SHSF.KeyGen}$, $\overline{C}$ denotes a negated circuit of C, and $r \xleftarrow{\$} \mathcal{Y}$. This change will go unnoticed due to the shift-hiding property of SHSF. Now, by the correctness of SHSF, we have $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x) = \mathsf{SHSF.Eval}(\mathsf{msk}_1, x) + \overline{C}(x) \cdot r$ for all x. In particular, the challenge output can be written as $\mathsf{SHSF.Eval}(\mathsf{msk}_1, x^*) + r$ since we must have $C(x^*) = 0$. On the other hand, for all inputs x such that $C(x) = 1$, we have $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x) = \mathsf{SHSF.Eval}(\mathsf{msk}_1, x)$. Since the constrained key K_C is an obfuscated program that computes $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x)$ for x such that $C(x) = 1$ and $\bot$ otherwise, the same functionality can be computed by using msk_1 instead of sk_1.

Thus, as a next game hop, we use the security of IO to hardwire msk_1 instead of sk_1 into the constrained key K_C. At this point, the constrained key K_C leaks no information of r since the distribution of msk_1 and r are independent. Thus, we can use the randomness of r to argue that the challenge output is independently uniform from the view of the adversary, which completes the security proof.

Next, we explain how to extend the above proof to the case of $O(1)$-collusion-resistance. A rough idea is to propagate a "masking term" (which was r in the single-key case) through a "chain" of secret keys of SHSF so that the masking term only appears in the challenge output and not used at all for generating constrained keys. We let C_j denote the j-th constrained key query. Then we consider the following game hops.

The first game hop is similar to the single-key case except for the choice of the shifting function. Specifically, we replace the master key $\mathsf{K} = \mathsf{sk}^{\mathsf{sim}}$ with a shifted secret key sk_1 that is generated by $\mathsf{SHSF.Shift}(\mathsf{msk}_1, \overline{C}_1(\cdot) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_2^{\mathsf{sim}}, \cdot))$ where msk_1 is a master key generated by $\mathsf{SHSF.KeyGen}$ and $\mathsf{sk}_2^{\mathsf{sim}}$ is another secret key generated by $\mathsf{SHSF.Sim}$. Similarly to the case of the single-key security, the way of generating K can be made dependent on the first constrained key query C_1 since K is needed for the first time when responding to the first constrained key query.[4] By the correctness of SHSF, we have $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x) = \mathsf{SHSF.Eval}(\mathsf{msk}_1, x) + \overline{C}_1(x) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_2^{\mathsf{sim}}, x)$ for all x. Especially, for all inputs x such that $C_1(x) = 1$, we have $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x) = \mathsf{SHSF.Eval}(\mathsf{msk}_1, x)$. Therefore, by using the security of IO, we can hardwire (msk_1, C_1) instead of $(\mathsf{K} = \mathsf{sk}_1, C_1)$ into the first constrained key K_{C_1} since it only evaluates the CPRF on x such that $C_1(x) = 1$. Here, note that we do *not* need to hard-wire the value $\mathsf{sk}_2^{\mathsf{sim}}$ in the first constrained key K_{C_1} since $\mathsf{SHSF.SEval}(\mathsf{sk}_2^{\mathsf{sim}}, x)$ part is canceled when $C_1(x) = 1$.

Similarly, for the j-th constrained key for $j \geq 2$, we hardwire $(\mathsf{msk}_1, \mathsf{sk}_2^{\mathsf{sim}}, C_1, C_j)$ instead of $(\mathsf{K} = \mathsf{sk}_1, C_j)$. We note that we have to hardwire $\mathsf{sk}_2^{\mathsf{sim}}$ and C_1 into these constrained keys since they may need to evaluate the CPRF on x such that $C_1(x) = 0$. At this point, the challenge value is $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x^*) =$

[4] Recall that we assume that an adversary does not make an evaluation query and that the challenge query is made at the end of the game.

$\mathsf{SHSF.Eval}(\mathsf{msk}_1, x^*) + \mathsf{SHSF.SEval}(\mathsf{sk}_2^{\mathsf{sim}}, x^*)$ where x^* denotes the challenge query since we must have $C_1(x^*) = 0$. Next, we apply similar game hops for the next secret key $\mathsf{sk}_2^{\mathsf{sim}}$. Specifically, we replace $\mathsf{sk}_2^{\mathsf{sim}}$ with sk_2 generated by $\mathsf{SHSF.Shift}(\mathsf{msk}_2, \overline{C}_2(\cdot) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_3^{\mathsf{sim}}, \cdot))$ where msk_2 is another master key generated by $\mathsf{SHSF.KeyGen}$ and $\mathsf{sk}_3^{\mathsf{sim}}$ is another secret key generated by $\mathsf{SHSF.Sim}$. Again, we remark that the way of generating sk_2 can be made dependent on C_2 since it is needed for the first time when responding to the second constrained key query. At this point, we only have to hardwire (msk_1, C_1) into the first constrained key, $(\mathsf{msk}_1, \mathsf{msk}_2, C_1, C_2)$ into the second constrained key, and $(\mathsf{msk}_1, \mathsf{msk}_2, \mathsf{sk}_3^{\mathsf{sim}}, C_1, C_2, C_j)$ into the j-th constrained key for $j \geq 3$, and the challenge output is $\mathsf{SHSF.Eval}(\mathsf{msk}_1, x^*) + \mathsf{SHSF.Eval}(\mathsf{msk}_2, x^*) + \mathsf{SHSF.SEval}(\mathsf{sk}_3^{\mathsf{sim}}, x^*)$. Repeating similar game hops Q times where Q is the number of constrained key queries, we eventually reach the game where

- for each $j \in [Q]$, $\{(\mathsf{msk}_i, C_i)\}_{i \in [j]}$ is hardwired into the j-th constrained key, and
- the challenge output is $\sum_{i \in [Q]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) + \mathsf{SHSF.SEval}(\mathsf{sk}_{Q+1}^{\mathsf{sim}}, x^*)$.

Especially, in this game, $\mathsf{sk}_{Q+1}^{\mathsf{sim}}$ is only used for generating the challenge output and independent of all constrained keys. Thus, we can conclude that the challenge output is random relying on the randomness of $\mathsf{sk}_{Q+1}^{\mathsf{sim}}$.[5] This completes the proof of the $O(1)$-collusion-resistant adaptive security of our CPRF.

At first glance, the above security proof may work even if an adversary makes (bounded) polynomial number of constrained keys since we only have to hardwire polynomial number of keys and circuits into constrained keys. However, the problem is that the size of the master key msk depends on the maximal size of the shifting function in the LWE-based construction of SHSF given in [39]. In our construction of CPRF, the corresponding shifting function for msk_i depends on sk_{i+1}, and thus msk_i must be polynomially larger than sk_{i+1}, which itself is larger than msk_{i+1}. Thus, the size of msk_i grows polynomially in each layer of the nest. This is the reason why our proof is limited to the $O(1)$-collusion-resistant case.

We leave it open to construct an SHSF whose master key size does not depend on the maximal size of the shifting function, which would result in a bounded polynomial collusion-resistant adaptively secure CPRF for $\mathsf{P/poly}$.

3 Preliminaries

Notations. For a distribution or random variable X, we write $x \xleftarrow{\$} X$ to denote the operation of sampling a random x according to X. For a set S, we write $s \xleftarrow{\$} S$ to denote the operation of sampling a random s from the uniform distribution over S. Let $U(S)$ denote the uniform distribution over the set S. For a prime q, we represent the elements in $\mathbb{Z}_q$ by integers in the range $[-(q-1)/2, (q-1)/2]$.

[5] We can show that $\mathsf{SHSF.SEval}(\mathsf{sk}^{\mathsf{sim}}, x)$ is uniformly distributed in $\mathcal{Y}$ over the choice of $\mathsf{sk}^{\mathsf{sim}}$ for any fixed x

For $2 \leq p < q$ and $x \in \mathbb{Z}_q$ (or $\mathbb{Z}$), we define $\lfloor x \rfloor_p := \lfloor (p/q) \cdot x \rfloor \in \mathbb{Z}_p$. We will represent vectors by bold-face letters, and matrices by bold-face capital letters. Unless stated otherwise, we will assume that all vectors are column vectors.

Gadget Matrix. Let $n, q \in \mathbb{Z}$ and $m \geq n\lceil \log q \rceil$. A gadget matrix $\mathbf{G}$ is defined as $\mathbf{I}_n \otimes (1, 2, ..., 2^{\lceil \log q \rceil - 1})$ padded with $m - n\lceil \log q \rceil$ zero columns. For any t, there exists an efficient deterministic algorithm $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times t} \to \{0,1\}^{m \times t}$ that takes $\mathbf{U} \in \mathbb{Z}_q^{n \times t}$ as input and outputs $\mathbf{V} \in \{0,1\}^{m \times t}$ such that $\mathbf{GV} = \mathbf{U}$.

3.1 Admissible Hash Functions and Matrix Embeddings

We prepare the definition of (balanced) admissible hash functions.

Definition 3.1. *Let $\ell := \ell(\kappa)$ and $n := n(\kappa)$ be integer valued polynomials. For $K \in \{0,1,\perp\}^\ell$, we define the* partitioning function $P_K : \{0,1\}^\ell \to \{0,1\}$ *as*

$$P_K(z) = \begin{cases} 0, & \text{if } (K_i = \perp) \vee (K_i = z_i) \\ 1, & \text{otherwise} \end{cases}$$

where K_i and z_i denote the i^{th} bit of K and z, respectively. We say that an efficiently computable function $\mathsf{H}_{\mathsf{adm}} : \{0,1\}^n \to \{0,1\}^\ell$ is a balanced admissible hash function*, if there exists a PPT algorithm $\mathsf{PrtSmp}(1^\kappa, Q(\kappa), \delta(\kappa))$, which takes as input a polynomially bounded function $Q := Q(\kappa)$ where $Q : \mathbb{N} \to \mathbb{N}$ and a noticeable function $\delta := \delta(\kappa)$ where $\delta : \mathbb{N} \to (0,1]$, and outputs $K \in \{0,1,\perp\}^\ell$ such that:*

1. *There exists $\kappa_0 \in \mathbb{N}$ such that*

$$\Pr\left[K \xleftarrow{\$} \mathsf{PrtSmp}\big(1^\kappa, Q(\kappa), \delta(\kappa)\big) : K \in \{0,1,\perp\}^\ell\right] = 1$$

   for all $\kappa > \kappa_0$. Here κ_0 may depend on the functions Q and δ.
2. *For $\kappa > \kappa_0$, there exists functions $\gamma_{\max}(\kappa)$ and $\gamma_{\min}(\kappa)$ that depend on functions Q and δ such that for all $x_1, \cdots, x_{Q(\kappa)}, x^* \in \{0,1\}^n$ with $x^* \notin \{x_1, \cdots, x_{Q(\kappa)}\}$,*

$$\gamma_{\min}(\kappa) \leq \Pr\left[P_K(\mathsf{H}_{\mathsf{adm}}(x_1)) = \cdots = P_K(\mathsf{H}_{\mathsf{adm}}(x_{Q(\kappa)})) = 1 \wedge P_K(\mathsf{H}_{\mathsf{adm}}(x^*)) = 0\right] \leq \gamma_{\max}(\kappa)$$

   holds and the function $\tau(\kappa)$ defined as

$$\tau(\kappa) := \gamma_{\min}(\kappa) \cdot \delta(\kappa) - \frac{\gamma_{\max}(\kappa) - \gamma_{\min}(\kappa)}{2}$$

   is noticeable. The probability is taken over the choice of $K \xleftarrow{\$} \mathsf{PrtSmp}(1^\kappa, Q(\kappa), \delta(\kappa))$.

Theorem 3.1 ([30], Theorem 1). *Let $n = \Theta(\kappa)$ and $\ell = \Theta(\kappa)$. If $\mathsf{H}_{\mathsf{adm}} : \{0,1\}^n \to \{0,1\}^\ell$ is a code with minimal distance $c \cdot \ell$ for a constant $c \in (0, 1/2]$, then $\mathsf{H}_{\mathsf{adm}}$ is a balanced admissible hash function. Specifically, there exists a PPT algorithm $\mathsf{PrtSmp}(1^\kappa, Q, \delta)$ which takes as input $Q \in \mathbb{N}$ and $\delta \in (0, 1]$ and outputs $K \in \{0, 1, \perp\}^\ell$ with η' components not equal to $\perp$, where*

$$\eta' = \left\lfloor \frac{\log(2Q + Q/\delta)}{-\log(1-c)} \right\rfloor \quad \text{and} \quad \gamma(\kappa) = 2^{-\eta'-1} \cdot \delta.$$

In particular, when $Q = \mathsf{poly}(\kappa)$ and $\delta = 1/\mathsf{poly}(\kappa)$, then $\eta' = O(\log \kappa)$ and $\gamma(\kappa) = 1/\mathsf{poly}(\kappa)$.

The following is taken from [11] and [43].

Lemma 3.1 (Compatible Algorithms with Partitioning Functions). *Let $P_K : \{0,1\}^\ell \to \{0,1\}$ be a partitioning function where $K \in \{0, 1, \perp\}^\ell$ and assume that K has at most $O(\log \kappa)$ entries in $\{0, 1\}$. Then, there exist deterministic PPT algorithms $(\mathsf{Encode}, \mathsf{PubEval}, \mathsf{TrapEval})$ with the following properties:*

- $\mathsf{Encode}(K)$: *on input K, it outputs $\mu \in \{0,1\}^u$ where $u = O(\log^2 \kappa)$,*
- $\mathsf{PubEval}(x, \mathbf{A})$: *on input $x \in \{0,1\}^\ell$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times mu}$, it outputs $\mathbf{A}_x \in \mathbb{Z}_q^{n \times m}$,*
- $\mathsf{TrapEval}(\mu, x, \mathbf{A}_0, \mathbf{R})$: *on input $\mu \in \{0,1\}^u$, $x \in \{0,1\}^\ell$, $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{R} \in \{-1, 0, 1\}^{m \times mu}$, it outputs $\mathbf{R}_x \in \mathbb{Z}^{m \times m}$,*
- *If $\mathbf{A} := \mathbf{A}_0\mathbf{R} + \mu \otimes \mathbf{G}$ and $\mathbf{R} \in \{0,1\}^{m \times mu}$ where μ is viewed as a row vector in $\{0,1\}^u$, then for $\mathbf{A}_x = \mathsf{PubEval}(x, \mathbf{A})$ and $\mathbf{R}_x = \mathsf{TrapEval}(\mu, x, \mathbf{A}, \mathbf{R})$, we have $\mathbf{A}_x = \mathbf{A}_0\mathbf{R}_x + (1 - P_K(x)) \cdot \mathbf{G}$ and $\|\mathbf{R}_x\|_\infty \leq m^3 u\ell$.*
- *Moreover, $\mathbf{R}_x$ can be expressed as $\mathbf{R}_x = \mathbf{R}_0 + \mathbf{R}'_x$ where $\mathbf{R}_0$ is the first m columns of $\mathbf{R}$ and is distributed independently from $\mathbf{R}'_x$.*

Remark 3.1. The last item is non-standard, however, we note that it is without loss of generality. This is because we can always satisfy the last condition by constructing a new $\mathsf{PubEval}'$ which simply samples one extra random matrix $\bar{\mathbf{R}}$ and adds $\mathbf{A}_0\bar{\mathbf{R}}$ to $\mathbf{A}_x = \mathsf{PubEval}(x, \mathbf{A})$. This requirement is only required in our security proof of our CPRF for inner product predicates. More details can be found in [35], Section 4.3.

The following lemma is taken from [31], and is implicit in [9,30,43].

Lemma 3.2 ([31], Lemma 8). *Let us consider a random variable $\mathsf{coin} \xleftarrow{\$} \{0,1\}$ and a distribution $\mathcal{D}$ that takes as input a bit $b \in \{0,1\}$ and outputs $(x, \widehat{\mathsf{coin}})$ such that $x \in \mathcal{X}$ and $\widehat{\mathsf{coin}} \in \{0,1\}$, where $\mathcal{X}$ is some domain. For $\mathcal{D}$, define ϵ as*

$$\epsilon := \left| \Pr\left[\mathsf{coin} \xleftarrow{\$} \{0,1\}, (x, \widehat{\mathsf{coin}}) \xleftarrow{\$} \mathcal{D}(\mathsf{coin}) : \mathsf{coin} = \widehat{\mathsf{coin}}\right] - \frac{1}{2} \right|.$$

Let γ be a map that maps an element in $\mathcal{X}$ to a value in $[0, 1]$. Let us further consider a modified distribution $\mathcal{D}'$ that takes as input a bit $b \in \{0,1\}$ and outputs $(x, \widehat{\mathsf{coin}})$. To sample from $\mathcal{D}'$, we first sample $(x, \widehat{\mathsf{coin}}) \xleftarrow{\$} \mathcal{D}(b)$, and then with

probability $1-\gamma(x)$*, we re-sample* $\widehat{\mathsf{coin}}$ *as* $\widehat{\mathsf{coin}} \xleftarrow{\$} \{0,1\}$*. Finally,* $\mathcal{D}'$ *outputs* $(x, \widehat{\mathsf{coin}})$*. Then, the following holds.*

$$\left|\Pr\left[\mathsf{coin} \xleftarrow{\$} \{0,1\}, (x, \widehat{\mathsf{coin}}) \xleftarrow{\$} \mathcal{D}'(\mathsf{coin}) : \mathsf{coin} = \widehat{\mathsf{coin}}\right] - \frac{1}{2}\right| \geq \gamma_{\min} \cdot \epsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2}$$

where $\gamma_{\min}$ *(resp.* $\gamma_{\max}$*) is the maximum (resp. minimum) of* $\gamma(x)$ *taken over all possible* $x \in \mathcal{X}$*.*

4 CPRFs for Bit-Fixing Predicates from Standard PRFs

In this section, we provide a construction of an adaptively pseudorandom on constrained points, Q-collusion resistant CPRFs for the bit-fixing predicate from any PRF, where Q can be set to be any constant independent of the security parameter. In particular, the result implies the existence of such CPRFs from one-way functions [25,27]. Recall that no other CPRFs are known to be adaptive and/or to achieve Q-collusion resistance for any $Q > 1$ both from the standard assumptions and in the standard model, excluding the CPRF for the trivial singleton sets $F = \{\{x\} \mid x \in \{0,1\}^n\}$ [15] or the selectively-secure and collusion-resistant CPRF for prefix-fixing predicates by [5].

Note that it is easy to extend our CPRF for the bit-fixing predicate to a CPRF for the t-CNF predicate where t is a constant. See the full version for the detail.

4.1 Preparation: Bit-Fixing Predicates

Here, we provide the constraint class we will be considering: bit-fixing predicates.

Definition 4.1 (Bit-Fixing Predicate). *For a vector* $v \in \{0,1,*\}^\ell$*, define the circuit* $C_v^{\mathsf{BF}} : \{0,1\}^\ell \to \{0,1\}$ *associated with* v *as*

$$C_v^{\mathsf{BF}}(x) = \bigwedge_{i=1}^{\ell} \Big((v_i \stackrel{?}{=} x_i) \bigvee (v_i \stackrel{?}{=} *) \Big),$$

where v_i *and* x_i *denote the* i^{th} *bit of the string* v *and* x*, respectively. Then, the family of bit-fixing predicates (with input length* ℓ*) is defined as*

$$\mathcal{C}_\ell^{\mathsf{BF}} := \{C_v^{\mathsf{BF}} \mid v \in \{0,1,*\}^\ell\}.$$

Since we can consider a canonical representation of the circuit C_v^{BF} given the string $v \in \{0,1,*\}^\ell$, with an abuse of notation, we may occasionally write $v \in \mathcal{C}_\ell^{\mathsf{BF}}$ and view v as C_v^{BF} when the meaning is clear.

We also define a helper function $G_{\mathsf{auth}}^{\mathsf{BF}}$ which, informally, outputs a set of all the authorized inputs corresponding to a bit-fixing predicate. For any $v \in \{0,1,*\}^\ell$ and $T = (t_1, \cdots, t_Q) \in [\ell]^Q$ such that $Q \leq \ell$, let us define $v_T \in \{0,1,*\}^Q$ as the

string $v_{t_1} v_{t_2} \cdots v_{t_Q}$, where v_i is the i^{th} bit of v. Then we define the function $G^{\mathsf{BF}}_{\mathsf{auth}}$ as follows.

$$G^{\mathsf{BF}}_{\mathsf{auth}}(v_T) = \{w \in \{0,1\}^Q \mid C^{\mathsf{BF}}_{v_T}(w) = 1\}.$$

In words, it is the set of all points with the same length as v_T that equals to v_T on the non-wild card entries. For example, if $\ell = 8, Q = 5$, $v = 011 * 01 * 1$, and $T = (4, 1, 2, 6, 1)$, then $v_T = *0110$ and the authorized set of points would be $G^{\mathsf{BF}}_{\mathsf{auth}}(v_T) = \{00110, 10110\}$. Here, with an abuse of notation, we define the function $G^{\mathsf{BF}}_{\mathsf{auth}}$ for all input lengths.

4.2 Construction

Let $n = n(\kappa)$, and $k = k(\kappa)$ be integer-valued positive polynomials of the security parameter κ and Q be any constant positive integer smaller than n. Let $\mathcal{C}^{\mathsf{BF}} := \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}} := \{\mathcal{C}^{\mathsf{BF}}_{n(\kappa)}\}_{\kappa\in\mathbb{N}}$ be a set of family of circuits representing the class of constraints. Let $\mathit{\Pi}_{\mathsf{PRF}} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ be any PRF with input length n and output length k.

Our Q-collusion resistance CPRF $\mathit{\Pi}_{\mathsf{CPRF}}$ for the constrained class $\mathcal{C}^{\mathsf{BF}}$ is provided as follows:

$\mathsf{CPRF.Gen}(1^\kappa)$: On input the security parameter 1^κ, it runs $\mathsf{K}_{T,w} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ and $\widehat{\mathsf{K}}_{T,w} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [n]^Q$ and $w \in \{0,1\}^Q$. Then it outputs the master key as

$$\mathsf{K} = \Big((\mathsf{K}_{T,w}), (\widehat{\mathsf{K}}_{T,w})\Big)_{T\in[n]^Q, w\in\{0,1\}^Q}.$$

$\mathsf{CPRF.Eval}(\mathsf{K}, x)$: On input the master key K and input $x \in \{0,1\}^n$, it first parses

$$\big((\mathsf{K}_{T,w}), (\widehat{\mathsf{K}}_{T,w})\big)_{T\in[n]^Q, w\in\{0,1\}^Q} \leftarrow \mathsf{K}.$$

It then computes

$$y = \bigoplus_{T\in[n]^Q} \mathsf{PRF.Eval}(\mathsf{K}_{T,x_T}, x),$$

where recall $x_T \in \{0,1\}^Q$ is defined as the string $x_{t_1} x_{t_2} \cdots x_{t_Q}$ and $T = (t_1, \cdots, t_Q)$. Finally, it outputs $y \in \{0,1\}^k$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C^{\mathsf{BF}}_v)$: On input the master key K and a circuit $C^{\mathsf{BF}}_v \in \mathcal{C}^{\mathsf{BF}}_n$, it first parses K into $\big((\mathsf{K}_{T,w}), (\widehat{\mathsf{K}}_{T,w})\big)_{T\in[n]^Q, w\in\{0,1\}^Q} \leftarrow \mathsf{K}$ and sets $v \in \{0,1,*\}^n$ as the representation of C^{BF}_v. Then it outputs the constrained key

$$\mathsf{K}_v = \Big(\widetilde{\mathsf{K}}_{T,w}\Big)_{T\in[n]^Q, w\in\{0,1\}^Q},$$

where $\widetilde{\mathsf{K}}_{T,w} = \mathsf{K}_{T,w}$ if $w \in G^{\mathsf{BF}}_{\mathsf{auth}}(v_T)$, and $\widetilde{\mathsf{K}}_{T,w} = \widehat{\mathsf{K}}_{T,w}$ otherwise. Recall that $G^{\mathsf{BF}}_{\mathsf{auth}}(v_T) = \{w \in \{0,1\}^Q \mid C^{\mathsf{BF}}_{v_T}(w) = 1\}$.

$\mathsf{CPRF.ConstrainEval}(\mathsf{K}_v, x)$: On input the constrained key K_v and an input $x \in \{0,1\}^n$, it first parses $\left(\widetilde{\mathsf{K}}_{T,w}\right)_{T\in[n]^Q, w\in\{0,1\}^Q} \leftarrow \mathsf{K}_v$. It then uses the PRF keys included in the constrained key and computes

$$y = \bigoplus_{T\in[n]^Q} \mathsf{PRF.Eval}(\widetilde{\mathsf{K}}_{T,x_T}, x).$$

Finally, it outputs $y \in \{0,1\}^k$.

Theorem 4.1. *If the underlying PRF Π_{PRF} is adaptively pseudorandom, then our above CPRF Π_{CPRF} for the bit-fixing predicate $\mathcal{C}^{\mathsf{BF}}$ is adaptively pseudorandom on constrained points and Q-collusion resistant for any $Q = O(1)$.*

We omit the proofs of correctness and security due the space limit. See the full version for omitted proofs.

5 CPRF for Inner Products

5.1 Construction

In this section, we construct CPRFs for inner products over the integer. Fix a security parameter κ and define the following quantities:

- Let $\mathcal{D} := [-B, B]^\ell \subset \mathbb{Z}^\ell$ where inner products between two vectors $\mathbf{v}, \mathbf{w} \in \mathcal{D}$ are defined in the natural way over the integers. Let $\mathcal{C}^{\mathsf{IP}} := \{C_{\mathbf{v}}\}_{\mathbf{v}\in\mathcal{D}}$ be the set of circuits where each $C_{\mathbf{v}} : \mathcal{D} \to \mathbb{Z}$ is defined as $C_{\mathbf{v}}(\mathbf{w}) = (\langle \mathbf{v}, \mathbf{w}\rangle \stackrel{?}{=} 0)$, that is, if the inner product is zero then it outputs 1, and otherwise 0.
- Let $\mathsf{bin} : \mathcal{D} \to \{0,1\}^{\hat{\ell}}$ be a one-to-one map which provides a binary representation of elements in $\mathcal{D}$ where $\hat{\ell} := \ell \cdot \lceil \log(2B+1) \rceil$.
- Let $\mathsf{H}_{\mathsf{adm}} : \{0,1\}^{\hat{\ell}} \to \{0,1\}^L$ be a balanced admissible hash function where $L = \Theta(\kappa)$ by Theorem 3.1.
- Let $\mathcal{H}_{\mathsf{wise}} = \{\mathsf{H}_{\mathsf{wise}} : \mathbb{Z}_p^m \to \mathbb{Z}_p^k\}$ be a family of ζ-wise independent hash functions.
- Let $n, m, u, q, p, \beta, \bar{\beta}$ be additional parameters used within the CPRF scheme and let $h, \alpha_{\mathsf{LWE}}, \alpha_1, \alpha_2$ be parameters used within the security proof, where h, α_{LWE} are LWE-related. The details on the parameters setting is provide after our construction below.

Our CPRF Π_{CPRF} for the constrained class of inner products over the integer $\mathcal{C}^{\mathsf{IP}}$ is provided below. Here, the domain, range, and key space of our CPRF are $\mathcal{D}$, $\mathbb{Z}_p^k$, and $\mathbb{Z}^{n\times\ell}$, respectively.

$\mathsf{CPRF.Setup}(1^\kappa)$: On input the security parameter 1^κ, it first samples random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times mu}$. It also samples a ζ-wise independent hash function $\mathsf{H}_{\mathsf{wise}} : \mathbb{Z}_p^m \to \mathbb{Z}_p^k$.

$$\mathsf{pp} = \left(\mathbf{A}, \mathsf{H}_{\mathsf{wise}}\right).$$

$\mathsf{CPRF.Gen}(\mathsf{pp})$: On input the public parameter pp, it samples a matrix $\mathbf{S} \xleftarrow{\$} [-\bar{\beta}, \bar{\beta}]^{n \times \ell} \subset \mathbb{Z}^{n \times \ell}$ and sets the master key as $\mathsf{K} = \mathbf{S}$.

$\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{K}, \mathbf{x})$: On input the public parameter pp, master key $\mathsf{K} = \mathbf{S} \in \mathbb{Z}^{n \times \ell}$ and input $\mathbf{x} \in \mathcal{D}$, it first computes $\mathbf{s} = \mathbf{S}\mathbf{x} \in \mathbb{Z}^n$ and $x = \mathsf{H}_{\mathsf{adm}}(\mathsf{bin}(\mathbf{x})) \in \{0,1\}^L$. It then computes

$$\mathbf{z} = \left\lfloor \mathbf{s}^\top \mathbf{A}_x \right\rceil_p \in \mathbb{Z}_p^m,$$

where $\mathbf{A}_x = \mathsf{PubEval}(x, \mathbf{A})$. Finally, it outputs $\mathbf{v} = \mathsf{H}_{\mathsf{wise}}(\mathbf{z}) \in \mathbb{Z}_p^k$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C_{\mathbf{y}})$: On input the master key $\mathsf{K} = \mathbf{S}$ and constraint $C_{\mathbf{y}} \in \mathcal{C}^{\mathsf{IP}}$, it first samples a random vector $\mathbf{d} \xleftarrow{\$} [-\beta, \beta]^n$. It then outputs constrained key $\mathsf{K}_{\mathbf{y}} \in \mathbb{Z}^{n \times \ell}$ defined as

$$\mathsf{K}_{\mathbf{y}} = \mathbf{S} + \mathbf{d} \otimes \mathbf{y}^\top.$$

$\mathsf{CPRF.ConstrainEval}(\mathsf{pp}, \mathsf{K}_{\mathbf{y}}, x)$: On input the public parameter pp, constrained key $\mathsf{K}_{\mathbf{y}} = \mathbf{S}_{\mathbf{y}} \in \mathbb{Z}^{n \times \ell}$ and input $\mathbf{x} \in \mathcal{D}$, it first computes $\mathbf{s}_{\mathbf{y}} = \mathbf{S}_{\mathbf{y}}\mathbf{x} \in \mathbb{Z}^n$ and $x = \mathsf{H}_{\mathsf{adm}}(\mathsf{bin}(\mathbf{x})) \in \{0,1\}^L$. It then computes

$$\mathbf{z}_{\mathbf{y}} = \left\lfloor \mathbf{s}_{\mathbf{y}}^\top \mathbf{A}_x \right\rceil_p \in \mathbb{Z}_p^m,$$

where $\mathbf{A}_x = \mathsf{PubEval}(x, \mathbf{A})$. Finally, it outputs $\mathbf{v} = \mathsf{H}_{\mathsf{wise}}(\mathbf{z}_{\mathbf{y}}) \in \mathbb{Z}_p^k$.

5.2 Correctness and Parameter Selection

Correctness. We check correctness of our CPRF. Let $C_{\mathbf{y}}$ be any inner-product predicate in $\mathcal{C}^{\mathsf{IP}}$. By construction when we evaluate with a constrained key $\mathsf{K}_{\mathbf{y}}$ on input $\mathbf{x}$ we have

$$\mathbf{z}_{\mathbf{y}} = \left\lfloor \mathbf{s}_{\mathbf{y}}^\top \mathbf{A}_x \right\rceil_p = \left\lfloor \left(\left(\mathbf{S} + \mathbf{d} \otimes \mathbf{y}^\top \right) \mathbf{x} \right)^\top \mathbf{A}_x \right\rceil_p = \left\lfloor \mathbf{s}^\top \mathbf{A}_x + \langle \mathbf{x}, \mathbf{y} \rangle \cdot \mathbf{d}^\top \mathbf{A}_x \right\rceil_p,$$

where $\mathbf{s} = \mathbf{S}\mathbf{x}$. Therefore, if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ over $\mathbb{Z}$, i.e., the input $\mathbf{x}$ satisfies the constraint $C_{\mathbf{y}}$, then the right hand side will equal to $\lfloor \mathbf{s}^\top \mathbf{A}_x \rceil_p$, which is exactly what is computed by algorithm $\mathsf{CPRF.Eval}$ using the master key K. Hence, the output value $\mathbf{v} = \mathsf{H}_{\mathsf{wise}}(\mathbf{z})$ is the same for both values computed by the master key K and constrained key $\mathsf{K}_{\mathbf{y}}$.

Parameter Selection. We summarize the relation which our parameters must satisfy below. Note that some parameters only show up during the security proof. See the full version for the reasons of these parameter choices.

- $m > (n+1)\log q + \omega(\log n)$
- $\alpha_{\mathsf{LWE}} q > 2\sqrt{h}$
- $\|\mathbf{x}^\top \mathbf{S}^\top \mathbf{E} \mathbf{R}_x\|_\infty \leq \alpha_1$ for all $\mathbf{x} \in \mathcal{D}$
- $\|\ell B \mathbf{d}^\top \mathbf{E} \mathbf{R}_x\|_\infty \leq \alpha_2$ for all $\mathbf{x} \in \mathcal{D}$

– $q = 2mpB^{\ell} \cdot (\alpha_1 + \alpha_2) \cdot \kappa^{\omega(1)}$ for all $\mathbf{x} \in \mathcal{D}$
– $\bar{\beta} \geq \beta B$ and $\bar{\beta} = n\ell\beta B \cdot \kappa^{\omega(1)}$
– $\bar{n} := n \cdot \log(2\beta + 1) - h \cdot \log q = \Omega(\kappa)$

Fix $\ell = \ell(\kappa)$, $B = B(\kappa)$, $h = h(\kappa)$, and $u = O(\log^2 \kappa)$, where ℓ and B defines the constraint space (i.e., the set of vectors $\mathcal{D}$), $h(\geq \kappa)$ defines the lattice dimension for the underlying LWE problem, and u is the parameter for the admissible hash (see Sec. 3.1). We assume without loss of generality that ℓ and h are polynomial in κ. Then, one way to set the parameters would be as follows:

$$\begin{array}{ccc} n = \ell h^{1.1}, & m = \ell^2 h^{1.2}, & q = 2^{\ell} \ell^{14} h^{7.6} B^{\ell+2} \kappa^{3 \log \kappa}, \\ p = 10, & \alpha_{\mathsf{LWE}} = 2\sqrt{h} \cdot q^{-1}, & \zeta = \bar{n} + \ell \cdot \log B, \\ \alpha_1 = \alpha_2 = \ell^{12} h^{6.4} B^2 \kappa^{2 \log \kappa}, & \beta = 1, & \bar{\beta} = \ell^2 h^{1.1} B \kappa^{\log \kappa}, \end{array}$$

where we set q to be the next largest prime. Above we use the simplifying argument that for any positive constant c, we have $\kappa^{0.1} = \omega(\log^c \kappa)$ and $\log \kappa = \omega(1)$ for sufficiently large $\kappa \in \mathbb{N}$ and set ζ according to the deterministic randomness extractor lemma by Dodis [23] (see the full version for the detail). The output space of our CPRF is $\{0,1\}^{\bar{n}} = \{0,1\}^{\Theta(\kappa)}$.

5.3 Security Proof

Theorem 5.1. *The above CPRF Π_{CPRF} for the inner product predicate $\mathcal{C}^{\mathsf{IP}}$ is adaptively single-key pseudorandom on constrained points against adversaries that make exactly one constrained key query, assuming hardness of the $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha_{\mathsf{LWE}}q}}$ problem.*

Remark 5.1. We note that we can assume the adversary makes *exactly* one constrained key query without loss of generality. This is a useful condition to assume to handle adversaries that make no constrained key query but queries $\mathbf{x}^* = 0$ as the target input at the challenge phase. The above assumption holds because we can generically add security against adversaries that make no evaluation query by simply xoring an evaluated value of a (standard) PRF. The details are as follows. We add the same (standard) PRF key k both in the master secret key and constrained key. When evaluating on input $\mathbf{x}$, we will also xor the value $\mathsf{PRF}(k, \mathbf{x})$. Therefore, in case no constraint queries are made, pseudorandomness of $\mathsf{PRF}(k, \mathbf{x})$ can be used instead since k is not revealed.

We omitted the proofs due to the space limit. See the full version for the proofs.

6 CPRF for P/poly

6.1 Shift-Hiding Shiftable Function

Here, we review the notion of shift-hiding shiftable function (SHSF) introduced by Peikert and Shiehian [39]. We note that our definition of correctness is slightly

different from theirs. Specifically, we need a statistical notion of correctness whereas they only considered a computational notion of correctness. Nonetheless, a simple variant of their SHSF also satisfies our definition of correctness as seen in Lemma 6.1.

A SHSF with input space $\{0,1\}^\ell$ and output space $\mathbb{Z}_q^m$ with a rounding modulus $p < q$ consists of a tuple of PPT algorithms $\Pi_{\mathsf{SHSF}} = (\mathsf{SHSF.KeyGen}, \mathsf{SHSF.Eval}, \mathsf{SHSF.Shift}, \mathsf{SHSF.SEval}, \mathsf{SHSF.Sim})$[6] where:

$\mathsf{SHSF.KeyGen}(1^\kappa, 1^\sigma) \to \mathsf{msk}$: The key generation algorithm takes as input the security parameter 1^κ and the circuit size parameter 1^σ, and outputs a master key msk.

$\mathsf{SHSF.Eval}(\mathsf{msk}, x) \to \mathbf{y}$: The evaluation algorithm takes as input a master key msk and an input $x \in \{0,1\}^\ell$, and outputs $\mathbf{y} \in \mathbb{Z}_q^m$.

$\mathsf{SHSF.Shift}(\mathsf{msk}, C) \to \mathsf{sk}_C$: The shift algorithm takes as input a master key msk and a circuit C that computes a shift function, and outputs a shifted secret key sk_C.

$\mathsf{SHSF.SEval}(\mathsf{sk}_C, x) \to \mathbf{y}$: The shifted evaluation algorithm takes as input a secret key sk_C and an input $x \in \{0,1\}^\ell$, and outputs $\mathbf{y} \in \mathbb{Z}_q^m$.

$\mathsf{SHSF.Sim}(1^\kappa, 1^\sigma) \to \mathsf{sk}$: The key simulation algorithm takes as input the security parameter 1^κ and the circuit size parameter 1^σ, and outputs a simulated secret key sk.

We require Π_{SHSF} to satisfy the following properties.

p-Rounded ϵ-Correctness.

For all $x \in \{0,1\}^\ell$, circuit $C : \{0,1\}^\ell \to \mathbb{Z}_q^m$ whose description size is at most σ, and $\mathbf{v} \in \mathbb{Z}_q^m$, we have

$$\Pr[\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_C, x) + \mathbf{v} \rfloor_p \neq \lfloor \mathsf{SHSF.Eval}(\mathsf{msk}, x) + C(x) + \mathbf{v} \rfloor_p] \leq \epsilon$$

where $\mathsf{msk} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^\sigma)$ and $\mathsf{sk}_C \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}, C)$.

Shift Hiding. We define the notion of *shift hiding* for SHSFs. Informally, we require that a shifted secret key sk_C does not reveal the corresponding shifting circuit C.

Formally, this security notion is defined by the following game between an adversary $\mathcal{A}$ and a challenger:

Key Query: At the beginning of the game, the adversary is given the security parameter 1^κ, the circuit size parameter 1^σ, and returns a circuit $C : \{0,1\}^\ell \to \mathbb{Z}_q^m$ whose description size is at most σ.

Key Generation: The challenger chooses a random bit $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. Then it generates sk as follows:

- If $\mathsf{coin} = 0$, it generates $\mathsf{msk} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^\sigma)$ and $\mathsf{sk} \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}, C)$.

[6] In the original definition of [39], there is an additional setup algorithm that generates a public parameter. We omit this algorithm since in general we can always include the public parameter in the secret key.

- If $\mathsf{coin} = 1$, it generates $\mathsf{sk} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^\sigma)$.

It returns sk to $\mathcal{A}$.

Guess: Eventually, $\mathcal{A}$ outputs $\widehat{\mathsf{coin}}$ as a guess for coin.

We say the adversary $\mathcal{A}$ wins the game if $\widehat{\mathsf{coin}} = \mathsf{coin}$.

Definition 6.1. *An SHSF Π_{SHSF} is said to be shift hiding if for all $\sigma = \mathsf{poly}(\kappa)$ and PPT adversary $\mathcal{A}$, $|\Pr[\mathcal{A}\ wins] - 1/2| = \mathsf{negl}(\kappa)$ holds.*

Lemma 6.1 ([39]). *If $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha}}$ is hard for $q = 2^{\mathsf{poly}(\kappa,\sigma)} \cdot \epsilon^{-1}$, $m = n\lfloor \log q \rfloor$ and $\alpha = \mathsf{poly}(n)$, then for any $\ell = \mathsf{poly}(\kappa)$, there exists an SHSF from $\{0,1\}^\ell$ to $\mathbb{Z}_q^m$ that is shift hiding and satisfies p-rounded ϵ-correctness for some divisor p of q such that $p < \epsilon q$.*

Proof. Shiehian and Peikert [39] proved that if $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha}}$ is hard for $q = 2^{\mathsf{poly}(\kappa,\sigma)}$, $m = n\lfloor \log q \rfloor$ and $\alpha = \mathsf{poly}(n)$, then there exists an SHSF that is shift hiding and satisfies "approximated correctness", where the latter states that

$$\left| \Big(\mathsf{SHSF.SEval}(\mathsf{sk}_C, x) - \big(\mathsf{SHSF.Eval}(\mathsf{msk}, x) + C(x) \big) \Big)_i \right| \leq B = \kappa^{\mathsf{poly}(\kappa)}$$

for all $i \in [m]$, where $(\mathbf{z})_i$ for any $\mathbf{z} \in \mathbb{Z}^m$ denotes the i-th entry of $\mathbf{z}$. This implies that for any $\mathbf{v} \in \mathbb{Z}_q^m$ and p that divides q, we have

$$\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_C, x) + \mathbf{v} \rfloor_p = \lfloor \mathsf{SHSF.Eval}(\mathsf{msk}, x) + C(x) + \mathbf{v} \rfloor_p$$

as long as we have, for all $i \in [m]$,

$$(\mathsf{SHSF.SEval}(\mathsf{sk}_C, x) + \mathbf{v})_i \notin \frac{q}{p}\mathbb{Z} + [-B, B]. \tag{1}$$

Let us now consider a slight modification of their SHSF where an additional random vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^m$ is included in both msk and sk_C.[7] The modified $\mathsf{SHSF.Eval}$ and $\mathsf{SHSF.SEval}$ will now add $\mathbf{r}$ to the original outputs, e.g., run $\mathsf{SHSF.Eval}$ of [39] and add $\mathbf{r}$ to the output. It is clear that this modification does not harm the shift hiding property. With this slightly modified variant, for any fixed $x \in \{0,1\}^\ell$ and C, $\mathsf{SHSF.SEval}(\mathsf{sk}_C, x)$ is uniformly distributed over $\mathbb{Z}_q^m$ where the randomness is taken over the choice of $\mathsf{msk} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^\sigma)$ and $\mathsf{sk}_C \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}, C)$. Then, the probability that Equation (1) does not hold is at most $\frac{2pB}{q}$ for each $i \in [m]$. By taking the union bound, the probability that there exists $i \in [m]$ such that Equation (1) does not hold is at most $\frac{2pBm}{q}$. By taking the parameters so that $\frac{2pBm}{q} \leq \epsilon$, the slightly modified SHSF satisfies p-rounded ϵ-correctness. □

[7] Note that the vector $\mathbf{r}$ is sampled in the key generation, and not relevant to the vector $\mathbf{v}$ that appeared above.

6.2 Construction of CPRF

Here, we give a construction of an adaptively secure CPRF for all polynomial-size circuits (i.e., P/poly) from SHSF and IO.

Preparation. Before describing our construction, we prove a general lemma that enables us to focus on adversaries that do not make any evaluation queries. Namely, if we call constrained key queries made before (resp. after) the challenge query *pre-challange (resp. post-challenge) constrained key queries*, then we have the following lemma.

Lemma 6.2. *If there exists a CPRF for* P/poly *that is adaptively secure against adversaries that make at most* Q_1 *pre-challenge constrained key queries,* Q_2 *post-challenge constrained key queries, and no evaluation query, then the CPRF is adaptively secure against all adversaries that make at most* $Q_1 - 1$ *pre-challenge constrained key queries,* Q_2 *post-challenge constrained key queries, and* $\mathsf{poly}(\kappa)$ *evaluation queries.*

Roughly speaking, the lemma follows by considering a no-evaluation query adversary $\mathcal{A}$ which queries its challenger for a constrained key for the "partitioning function" ([30,43]). Then, $\mathcal{A}$ can simulate the view to the standard CPRF adversary $\mathcal{B}$ by simulating all evaluation queries made by $\mathcal{B}$ with this constrained key. In particular, with non-negligible probability, the partitioning function will output 1 for all evaluation queries and will output 0 for the challenge query. Therefore, $\mathcal{A}$ will be able to answer the evaluation queries made by $\mathcal{B}$ using its constrained key while it will not be able to answer the challenge query. Hence, with one extra constrained key query on the partitioning function, all evaluation queries can be simulated, which eliminates the necessity of evaluation queries. The full proof can be found in the full version.

Construction. Here, we construct an adaptively secure CPRF that tolerates $Q_1 = O(1)$ pre-challenge constrained key queries and $Q_2 = \mathsf{poly}(\kappa)$ post-challenge constrained key queries. By Lemma 6.2, we can assume that $\mathcal{A}$ does not make an evaluation query without loss of generality. Let z be the maximum description size of the circuit that is supported by our CPRF. Let $\Pi_{\mathsf{SHSF}} = (\mathsf{SHSF.KeyGen}, \mathsf{SHSF.Eval}, \mathsf{SHSF.Shift}, \mathsf{SHSF.SEval}, \mathsf{SHSF.Sim})$ be an SHSF with input space $\{0,1\}^\ell$ and output space $\mathbb{Z}_q^m$ that is shift hiding with a rounding modulus $p < \mathsf{negl}(\kappa) \cdot q$ that satisfies p-rounded ϵ-correctness where $\epsilon := 2^{-\ell}\mathsf{negl}(\kappa)$. We define parameters $\sigma_{Q_1+1}, ..., \sigma_1$ in the following recursive way.[8]

1. Set σ_{Q_1+1} as the maximum size of the circuit in the set $\{C_{\mathsf{eq}}[x^*, \mathbf{r}] \mid x^* \in \{0,1\}^\ell, \mathbf{r} \in \mathbb{Z}_q^m\}$, where $C_{\mathsf{eq}}[x^*, \mathbf{r}](\cdot)$ is a circuit which outputs $\mathbf{r}$ on input $x = x^*$, and 0 otherwise.[9]

[8] In the actual scheme, only σ_1 will appear and $\sigma_2, ..., \sigma_{Q_1+1}$ are only used in the security proof.

[9] Although there may be many ways to describe the circuit $C_{\mathsf{eq}}[x^*, \mathbf{r}]$, we consider the most obvious and standard one.

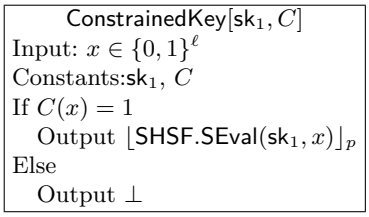

Fig. 3. Description of Program $\mathsf{ConstrainedKey}[\mathsf{sk}_1, C]$

2. For $i = Q_1, ..., 1$, set σ_i as the maximum size of the circuit that computes $\overline{C}(\cdot) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{i+1}, \cdot)$, where the max is taken over all $\mathsf{sk}_{i+1} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^{\sigma_{i+1}})$ and circuit $C : \{0,1\}^\ell \to \{0,1\}$ with description size at most z. Here, $\overline{C}$ denotes a circuit such that $\overline{C}(x) := (1 - C(x))$ for all $x \in \{0,1\}^\ell$ and $\overline{C}(\cdot) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{i+1}, \cdot)$ denotes the circuit that takes $x \in \{0,1\}^\ell$ as input and returns $\overline{C}(x) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{i+1}, x)$.

Note that the size of parameters satisfy $\sigma_1 > \sigma_2 > \cdots > \sigma_{Q_1+1}$.

Whenever we use IO, the circuit to be obfuscated is supposed to be padded so that they are as large as any circuit that replaces the circuit in the security proof. Then our CPRF is described as follows:[10]

$\mathsf{CPRF.Gen}(1^\kappa)$: On input the security parameter 1^κ, it generates $\mathsf{sk}_1 \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^{\sigma_1})$, and outputs $\mathsf{K} := \mathsf{sk}_1$.

$\mathsf{CPRF.Eval}(\mathsf{K}, x)$: On input the master key $\mathsf{K} = \mathsf{sk}_1$ and input $x \in \{0,1\}^\ell$, it computes $\mathbf{y} := \mathsf{SHSF.SEval}(\mathsf{sk}_1, x)$ and outputs $\lfloor \mathbf{y} \rfloor_p$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C)$: On input the master key $\mathsf{K} = \mathsf{sk}_1$ and constraint C, it returns $\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey}[\mathsf{sk}_1, C])$ where $\mathsf{ConstrainedKey}[\mathsf{sk}_1, C]$ is a program described in Figure 3 (with an appropriate padding).

$\mathsf{CPRF.ConstrainEval}(\mathsf{pp}, \mathsf{K}, x)$: On input the public parameter pp, constrained key K_C and input $x \in \{0,1\}^\ell$, it outputs $\mathsf{K}_C(x)$.

The following theorem addresses security of the above CPRF.

Theorem 6.1. *If* iO *is a secure indistinguishability obfuscator and* Π_{SHSF} *satisfies p-rounded* ϵ*-correctness and the shift hiding, then the above CPRF is adaptively secure against adversaries that make at most* $Q_1 = O(1)$ *pre-challenge constrained key queries,* $Q_2 = \mathsf{poly}(\kappa)$ *post-challenge constrained key queries, and no evaluation query.*

Combining this theorem with Lemmata 6.1 and 6.2 we obtain the following theorem.

Theorem 6.2. *If* $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha}}$ *is hard for* $n = \mathsf{poly}(\kappa)$, $q = 2^{\mathsf{poly}(\kappa,z)+\ell}$, $m = n\lfloor \log q \rfloor$, *and* $\alpha = \mathsf{poly}(n)$, , *then there exists a CPRF for* $\mathsf{P/poly}$ *that is adaptively*

[10] In our scheme, a public parameter is just the security parameter. So we omit the setup algorithm $\mathsf{CPRF.Setup}$.

secure against adversaries that make at most $O(1)$ pre-challenge constrained key queries, $\mathsf{poly}(\kappa)$ *post-challenge constrained key queries, and* $\mathsf{poly}(\kappa)$ *evaluation queries. Especially, under the same assumption, there exists an $O(1)$-collusion-resistant adaptively secure CPRF for* P/poly.

We omit the proof of Theorem 6.1 due to the space limit. See the full version for the proof.

References

1. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010.
2. J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Heidelberg, Aug. 2013.
3. N. Attrapadung, T. Matsuda, R. Nishimaki, S. Yamada, and T. Yamakawa. Constrained PRFs for NC^1 in traditional groups. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 543–574. Springer, Heidelberg, Aug. 2018.
4. N. Attrapadung, T. Matsuda, R. Nishimaki, S. Yamada, and T. Yamakawa. Adaptively single-key secure constrained PRFs for NC^1. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 223–253. Springer, Heidelberg, Apr. 2019.
5. A. Banerjee, G. Fuchsbauer, C. Peikert, K. Pietrzak, and S. Stevens. Key-homomorphic constrained pseudorandom functions. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 31–60. Springer, Heidelberg, Mar. 2015.
6. A. Banerjee and C. Peikert. New and improved key-homomorphic pseudorandom functions. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Heidelberg, Aug. 2014.
7. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, Apr. 2012.
8. M. Bellare, E. Kiltz, C. Peikert, and B. Waters. Identity-based (lossy) trapdoor functions and applications. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 228–245. Springer, Heidelberg, Apr. 2012.
9. M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 407–424. Springer, Heidelberg, Apr. 2009.
10. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, Aug. 2004.
11. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In P. Q. Nguyen and E. Oswald, editors,

EUROCRYPT 2014, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.
12. D. Boneh, S. Kim, and H. W. Montgomery. Private puncturable PRFs from standard lattice assumptions. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 415–445. Springer, Heidelberg, Apr. / May 2017.
13. D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan. Key homomorphic PRFs and their applications. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, Aug. 2013.
14. D. Boneh, K. Lewi, and D. J. Wu. Constraining pseudorandom functions privately. In S. Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 494–524. Springer, Heidelberg, Mar. 2017.
15. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, Dec. 2013.
16. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, Aug. 2014.
17. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, Mar. 2014.
18. Z. Brakerski, R. Tsabary, V. Vaikuntanathan, and H. Wee. Private constrained PRFs (and more) from LWE. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Heidelberg, Nov. 2017.
19. Z. Brakerski and V. Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, Mar. 2015.
20. R. Canetti and Y. Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 446–476. Springer, Heidelberg, Apr. / May 2017.
21. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010.
22. Y. Chen, V. Vaikuntanathan, and H. Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, Aug. 2018.
23. Y. Dodis. *Exposure-resilient cryptography*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.
24. G. Fuchsbauer, M. Konstantinov, K. Pietrzak, and V. Rao. Adaptive security of constrained PRFs. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, Dec. 2014.
25. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
26. S. Goldwasser, Y. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. *ICS*, pages 230–240, 2010.
27. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

28. D. Hofheinz, A. Kamath, V. Koppula, and B. Waters. Adaptively secure constrained pseudorandom functions. In I. Goldberg and T. Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 357–376. Springer, Heidelberg, Feb. 2019.
29. S. Hohenberger, V. Koppula, and B. Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 79–102. Springer, Heidelberg, Nov. / Dec. 2015.
30. T. Jager. Verifiable random functions from weaker assumptions. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 121–143. Springer, Heidelberg, Mar. 2015.
31. S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Heidelberg, Dec. 2016.
32. S. Katsumata and S. Yamada. Non-zero inner product encryption schemes from various assumptions: LWE, DDH and DCR. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 158–188. Springer, Heidelberg, Apr. 2019.
33. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, Nov. 2013.
34. B. Libert, A. Sakzad, D. Stehlé, and R. Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 332–364. Springer, Heidelberg, Aug. 2017.
35. B. Libert, D. Stehlé, and R. Titiu. Adaptively secure distributed PRFs from LWE. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 391–421. Springer, Heidelberg, Nov. 2018.
36. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, Apr. 2012.
37. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.
38. M. Naor, O. Reingold, and A. Rosen. Pseudorandom functions and factoring. *SIAM J. Comput.*, 31(5):1383–1404, 2002.
39. C. Peikert and S. Shiehian. Privately constraining and programming PRFs, the LWE way. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 675–701. Springer, Heidelberg, Mar. 2018.
40. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
41. R. Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Heidelberg, Aug. 2019.
42. B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.
43. S. Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 161–193. Springer, Heidelberg, Aug. 2017.