Adaptively Secure Constrained Pseudorandom Functions in the Standard Model*

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

[1]ISG, Royal Holloway University of London, UK
alex.davidson.2014@rhul.ac.uk
[2]National Institute of Advanced Industrial Science and Technology (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

February 5, 2020

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 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.

[*] This work is a merged version of [DN18] and [KY18] with many additional results.

[†] Part of this work was completed while the author undertook a research internship at NTT. 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).

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., [GGM86, NRR02, NR04, BPR12].

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 [BW13] alongside the concurrent works of Kiayias et al. [KPTZ13] and Boyle et al. [BGI14]. 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 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 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. 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 [BW13]. 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 Appendix A). 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 [BFP$^+$15, BV15, BTVW17, CC17, AMN$^+$18, CVW18, PS18], all constructions only achieve the weaker notion of selective security. In addition, other than selectively-secure CPRFs for the very restricted class of prefix-fixing predicates [BFP$^+$15], 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 P/poly, it is adaptive secure, and it is poly-collusion resistant) is only known based on IO in the ROM [HKKW19]. 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: [HKW15] only supports the very limited class of puncturing

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

predicates; [BLW17] only achieves selective security; and [AMN$^+$19] only achieves single-key security for the 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* 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) [BW13, BLW17, CC17, AMN$^+$18]. 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 Appendix C.) All other lattice-based CPRFs supporting beyond inner-product predicates (NC^1 or P/poly) [BV15, BTVW17, CC17, CVW18, PS18] achieve only selective security and admits no collusion-resistance.

3. We construct an adaptively secure and $O(1)$-collusion-resistant CPRF for P/poly from IO and the LWE assumption. More specifically, we use IO and shift-hiding shiftable functions [PS18], where the latter can be instantiated from the LWE assumption. This is the first adaptively secure CPRF for the class of 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 [HKW15]; achieves only selective security [BLW17]; or only achieves single-key security for the limited class of $\mathbf{NC}^1$ predicates [AMN$^+$19].

We also note that our first two constructions satisfy (weak) 1-key privacy, previously coined by Boneh et al. [BLW17] (see Remark 3.18 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 [Tsa19]. Other than identity-based encryption [ABB10, CHKP10] and non-zero inner product encryption [KY19], 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 [BW13], 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. [FKPR14]. One may wonder how our adaptively secure CPRF relates to the lower bound of adaptively secure CPRFs proven by Fuchsbauer et al. [FKPR14]. 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* [BW13]. 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 Appendix A. 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 CRPF, the main advantage is that it achieves adaptive security and supports the broadest predicate class P/poly without resorting to the ROM. Compared to the recent CPRF by Attrapadung et al. [AMN$^+$19], we provide a strict improvement since our first construction supports $O(1)$-collusion-resistance.

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
[BW13]	✓	poly	poly	LR	BDDH & ROM
[HKKW19]	✓	poly	0	P/poly	IO & ROM
[HKW15]	✓	poly	0	Puncturing	SGH & IO
[AMN$^+$18]	✓	1	1	BF	ROM
	✓	1	0	NC^1	L-DDHI & ROM
[AMN$^+$19]	✓	1	0	NC^1	SGH & IO
Section 4	✓	$O(1)$	$1^{\ddagger}$	t-CNF ($\supseteq$ BF)	OWF
Section 5	✓	1	$1^{\ddagger}$	IP	LWE
Section 6	✓	$O(1)$	0	P/poly	LWE & IO

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 [Tsa19] 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 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 [PS18] 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.

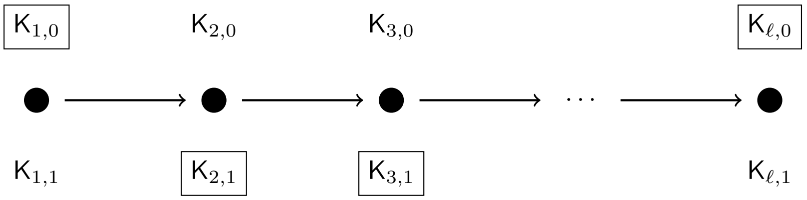

Figure 1: Length-ℓ directed line representation where each nodes are labeled with two PRF keys. In the figure, the choices of PRF keys correspond to some input $x = 011 \cdots 0$.

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 [GGM86, HILL99], 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 $\mathsf{K} = \{\mathsf{K}_{(i,j),(b_1,b_2)}\}_{(i,j)\in[\ell]^2,(b_1,b_2)\in\{0,1\}^2}$.

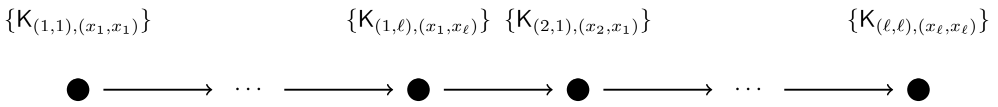
$\{\mathsf{K}_{(1,1),(x_1,x_1)}\}$
$\{\mathsf{K}_{(1,\ell),(x_1,x_\ell)}\}$
$\{\mathsf{K}_{(2,1),(x_2,x_1)}\}$
$\{\mathsf{K}_{(\ell,\ell),(x_\ell,x_\ell)}\}$
$\cdots$
$\cdots$

Figure 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 $(i_1, \dots, i_Q) \in [\ell]^Q$. Then we evaluate the CPRF on $x \in \{0,1\}^\ell$ as the output of:

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

The constraining algorithm works for a bit-fixing predicate defined by $v \in \{0,1,*\}^\ell$ by providing all keys $\mathsf{K}_{(i_1,\dots,i_Q),(b_1,\dots,b_Q)}$ 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 $v^{(1)}, \dots, v^{(Q)}$ and any constrained input x^*, there must exist a vector $(i'_1, \dots, i'_Q)$ such that $(x^*_{i'_j} \neq v^{(j)}_{i'_j}) \wedge (v^{(j)}_{i'_j} \neq *)$ for all $j \in [Q]$. Therefore, the key $\mathsf{K}_{(i'_1,\dots,i'_Q),(x^*_{i'_1},\dots,x^*_{i'_Q})}$ 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 4.2.

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 $(i_1, \dots, i_Q)$, 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 Appendix A, Table 2 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 which is addressed in Lemma 4.3 due to the non-trivial abort condition. Similar problems were identified by Waters [Wat05] 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 [BLMR13, BP14]. 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 $\mathbf{A_x} \in \mathbb{Z}_q^{n\times m}$ related to input $\mathbf{x}$ and simply outputs the value $\lfloor \mathbf{s}^\top \mathbf{A_x} \rceil_p \in \mathbb{Z}_p^m$, where $\lfloor a \rceil_p$ 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 $\mathbf{s}^\top \mathbf{A_x} + \mathsf{noise}$ 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 [BV15], many constructions of CPRF [BKM17, BTVW17] have built on top of the PRF of [BLMR13, BP14]. 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 [MP12]. 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 [BGG+14] 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 [BGG+14]. 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 [BLMR13, BP14] 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 [BLMR13, BP14], 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 $\hat{\mathbf{S}} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times \ell}$ and $\mathbf{d} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$; 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

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

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 [BLMR13, BP14] with secret key $\mathbf{d} \in \mathbb{Z}_q^n$.[2] 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 $\mathcal{A}$.

[2] 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.

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

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

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 [BGG+14], $\mathbf{A}_{\mathbf{x}}$ 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 [GKPV10, BKPW12, AKPW13, LSSS17]. 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 [LST18] that shows that this lossy LWE mode can be combined with homomorphic computation of [BGG+14] 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 [BB04]. 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 [LST18], there are subtle issues on why we have to resort to *deterministic* randomness extractors and not any randomness extractors. For more detail, see Section 5.

2.3 CPRF for P/poly

Our CPRF for P/poly is constructed based on IO and shift-hiding shiftable functions (SHSF).

First, we briefly recall SHSF. An SHSF consists of the following algorithms: a key generation algorithm SHSF.KeyGen, which generates a master key msk; an evaluation algorithm SHSF.Eval, which takes msk and $x \in \mathcal{X}$ as input and outputs $y \in \mathcal{Y}$; a shifting algorithm 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 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 CPRFs for 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 6.3 and its proof in Appendix D 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 $\perp$ 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 [SW14, BZ14]. 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 $\perp$ 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.[3] 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,

[3] 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.

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 $\mathsf{SHSF.SEval}(\mathsf{sk}_1, x^*) = \mathsf{SHSF.Eval}(\mathsf{msk}_1, x^*) + \mathsf{SHSF.SEval}(\mathsf{sk}_2^{\mathsf{sim}}, x^*)$ 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 $\mathsf{sk}_{Q+1}^{\mathsf{sim}}$.[4] 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 [PS18]. 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 $[-(q-1)/2, (q-1)/2]$. For $2 \leq p < q$ and $x \in \mathbb{Z}_q$ (or $\mathbb{Z}$), we define $\lfloor x \rceil_p := \lfloor (p/q) \cdot x \rceil \in \mathbb{Z}_p$. 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.

3.1 Lattices

Distributions. For an integer $m > 0$, let $D_{\mathbb{Z}^m, \sigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter $\sigma > 0$. We use the following lemmas regarding distributions.

Lemma 3.1 ([Reg05], Lemma 2.5). *We have* $\Pr[\|\mathbf{x}\| > \sigma\sqrt{m} : \mathbf{x} \leftarrow D_{\mathbb{Z}^m, \sigma}] < 2^{-2m}$.

[4] 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

Lemma 3.2 ([LST18], Lemma 2.3). *Let q be a prime and $\mathcal{B}$ be a distribution over $\mathbb{Z}_q^{n\times m}$ such that the statistical distance between $\mathcal{B}$ and $U(\mathbb{Z}_q^{n\times m})$ is less than ϵ. Then, for any distribution over $\mathcal{V}$ over $\mathbb{Z}_q^n$, if we sample $\mathbf{B} \xleftarrow{\$} \mathcal{B}$ and $\mathbf{v} \xleftarrow{\$} \mathcal{V}$, then $\mathbf{Bv}$ is distributed $\epsilon + \alpha \cdot (1 - 1/q^m)$-close to $U(\mathbb{Z}_q^n)$ where $\alpha := \Pr[\mathbf{v} = \mathbf{0}]$.*

Lemma 3.3 ([AKPW13], Lemma 2.7). *Let p, q be positive integers such that $p < q$. Given positive integers τ_1 and τ_2, the probability that there exists $e_1 \in [-\tau_1, \tau_1]$ and $e_2 \in [-\tau_2, \tau_2]$ such that $\lfloor a + e_1 \rceil_p \neq \lfloor a + e_2 \rceil_p$ for $a \xleftarrow{\$} \mathbb{Z}_q$ is smaller than $\frac{2(\tau_1+\tau_2)p}{q}$.*

We note that the above lemma is stated slightly different from the one in [AKPW13]. However, it is a direct consequence of the original proof.

Lemma 3.4 (Leftover Hash Lemma). *Let $q > 2$ be a prime, m, n, k be positive integers such that $m > (n+1)\log q + \omega(\log n)$, $k = \mathsf{poly}(n)$. Then, if we sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} \xleftarrow{\$} \{-1, 0, 1\}^{m\times k}$, then $(\mathbf{A}, \mathbf{AR})$ is distributed negligibly close to $U(\mathbb{Z}_q^{n\times m}) \times U(\mathbb{Z}_q^{n\times k})$.*

Hardness Assumption. We define the Learning with Errors (LWE) problem introduced by Regev [Reg05].

Definition 3.5 (Learning with Errors). *For integers n, m, a prime $q > 2$, an error distribution χ over $\mathbb{Z}$, and a PPT algorithm $\mathcal{A}$, the advantage for the* learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ *of $\mathcal{A}$ is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}} = \left| \Pr\left[\mathcal{A}\big(\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{z}^\top\big) = 1\right] - \Pr\left[\mathcal{A}\big(\mathbf{A}, \mathbf{b}^\top\big) = 1\right] \right|$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{b} \leftarrow \mathbb{Z}_q^m$, $\mathbf{z} \leftarrow \chi^m$. We say that the LWE *assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}}$ is negligible for all PPT algorithm $\mathcal{A}$.*

The (decisional) $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$ for $\alpha q > 2\sqrt{n}$ has been shown by Regev [Reg05] via a quantum reduction to be as hard as approximating the worst-case SIVP and GapSVP problems to within $\tilde{O}(n/\alpha)$ factors in the ℓ_2-norm in the worst case. In the subsequent works, (partial) dequantumization of the reduction were achieved [Pei09, BLP+13].

3.2 Admissible Hash Functions and Matrix Embeddings

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

Definition 3.6. *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*

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

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 3.7 ([Jag15], 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 $K \in \{0,1,\perp\}$ 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 [BGG$^+$14] and [Yam17].

Lemma 3.8 (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 3.9. 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 [LST18], Section 4.3.

The following lemma is taken from [KY16], and is implicit in [BR09, Jag15, Yam17].

Lemma 3.10 ([KY16], 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 $x \in \mathcal{X}$.

3.3 Deterministic Randomness Extractors

Lemma 3.11 ([Dod00], Corollary 3). *Fix any integers $\bar{n}, m$, and M, any real $\epsilon < 1$ and any collection $\mathcal{X}$ of M distributions over $\{0,1\}^m$ of min-entropy $\bar{n}$ each. Define*

$$\zeta = \bar{n} + \log M, \quad k = \bar{n} - \Big(2\log\frac{1}{\epsilon} + \log\log M + \log \bar{n} + O(1) \Big),$$

and let $\mathcal{F}$ be any family of ζ-wise independent functions from m bits to k bits. Then, with probability at least $(1-1/M)$, for any distribution $X \in \mathcal{X}$, the statistical distance between the distributions $f(X)$ for $f \xleftarrow{\$} \mathcal{F}$ and $U(\{0,1\}^k)$ are at most ϵ. Namely, with probability at least $(1-1/M)$, a random function f is a good deterministic extractor for the collection $\mathcal{X}$.

3.4 Indistinguishability Obfuscation

Here, we recall the definition of indistinguishability obfuscation (IO) (for all circuits) [BGI+12, GGH+16].

Definition 3.12 (Indistinguishability Obfuscation). *We say that a PPT algorithm* iO *is a secure* indistinguishability obfuscator *(IO), if it satisfies the following properties:*

Functionality: iO *takes a security parameter 1^λ and a circuit C as input, and outputs an obfuscated circuit $\widehat{C}$ that computes the same function as C. (We may drop 1^λ from an input to* iO *when λ is clear from the context.)*

Security: *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\left| \Pr\left[\begin{array}{l} (C_0, C_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(1^\lambda);\ \mathsf{coin} \leftarrow \{0,1\}; \\ \widehat{C} \xleftarrow{\$} \mathsf{iO}(1^\lambda, C_b);\ \widehat{\mathsf{coin}} \xleftarrow{\$} \mathcal{A}_2(\mathsf{st}, \widehat{C}) \end{array} : \widehat{\mathsf{coin}} = \mathsf{coin} \right] - \frac{1}{2} \right| = \mathsf{negl}(\kappa)$$

where it is required that C_0 and C_1 compute the same function and have the same description size.

3.5 Pseudorandom Functions

We first define the standard notion of pseudorandom functions (PRFs).

Syntax. Let $n = n(\kappa)$, and $k = k(\kappa)$ be integer-valued positive polynomials of the security parameter κ. A pseudorandom function is defined by a pair of PPT algorithms $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ where:

$\mathsf{PRF.Gen}(1^\kappa) \to \mathsf{K}$: The key generation algorithm takes as input the security parameter 1^κ and outputs a key $\mathsf{K} \in \{0,1\}^\kappa$.

$\mathsf{PRF.Eval}(\mathsf{K}, x) :\to y$: The evaluation algorithm takes as input $x \in \{0,1\}^n$ and outputs $y \in \{0,1\}^k$.

Pseudorandomness. We define the notion of (adaptive) *pseudorandomness* for the PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ using the following game between an adversary $\mathcal{A}$ and a challenger:

Setup: At the beginning of the game, the challenger prepares the key $\mathsf{K} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ and a set S initially set to be empty.

Evaluation Queries: During the game, $\mathcal{A}$ can adaptively query an evaluation on any input. When $\mathcal{A}$ submits $x \in \{0,1\}^n$ to the challenger, the challenger evaluates $y \xleftarrow{\$} \mathsf{PRF.Eval}(\mathsf{K}, x)$ and returns $y \in \{0,1\}^k$ to $\mathcal{A}$. It then updates $S \leftarrow S \cup \{x\}$.

Challenge Phase: At some point, $\mathcal{A}$ chooses its target input $x^\dagger \in \{0,1\}^n$ such that $x^\dagger \notin S$ and submits it to the challenger. The challenger chooses a random bit $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. If $\mathsf{coin} = 0$, it evaluates $y^\dagger \xleftarrow{\$} \mathsf{PRF.Eval}(\mathsf{K}, x^\dagger)$. If $\mathsf{coin} = 1$, it samples a random value $y^\dagger \xleftarrow{\$} \{0,1\}^k$. Finally, it returns $y^\dagger$ to $\mathcal{A}$.

Evaluation Queries: After the challenge phase, $\mathcal{A}$ may continue to make evaluation queries with the added restriction that it cannot query $x^\dagger$.

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 3.13. *A PRF Π_{PRF} is said to be (adaptively) pseudorandom if for all PPT adversary $\mathcal{A}$, the probability of $\mathcal{A}$ winning the above game is negligible.*

It is a well known fact that PRFs can be built entirely from one-way functions [GGM86, HILL99].

3.6 Constrained Pseudorandom Functions

We define constrained pseudorandom functions (CPRFs).

Syntax. Let $\mathcal{R} = \{\mathcal{R}_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{D} = \{\mathcal{D}_\kappa\}_{\kappa \in \mathbb{N}}$ be families of sets representing the range and domain of the PRF, respectively. Let $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of sets of the PRF keys. Finally, let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of circuits, where $\mathcal{C}_\kappa$ is a set of circuits with domain $\mathcal{D}_\kappa$ and range $\{0,1\}$ whose sizes are polynomially bounded.[5] In the following we drop the subscript when it is clear.

A constrained pseudorandom function for $\mathcal{C}$ is defined by the five PPT algorithms $\Pi_{\mathsf{CPRF}} = (\mathsf{CPRF.Setup}, \mathsf{CPRF.Gen}, \mathsf{CPRF.Eval}, \mathsf{CPRF.Constrain}, \mathsf{CPRF.ConstrainEval})$ where:

$\mathsf{CPRF.Setup}(1^\kappa) \to \mathsf{pp}$: The setup algorithm takes as input the security parameter 1^κ and outputs a public parameter pp.

$\mathsf{CPRF.Gen}(\mathsf{pp}) \to \mathsf{K}$: The key generation algorithm takes as input a public parameter pp and outputs a master key $\mathsf{K} \in \mathcal{K}$.

$\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{K}, x) :\to y$: The evaluation algorithm takes as input a public parameter pp, a master key K, and input $x \in \mathcal{D}$ and outputs $y \in \mathcal{R}$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C) :\to \mathsf{K}_C$: The constrained key generation algorithm takes as input a master key K and a circuit $C \in \mathcal{C}$ specifying the constraint and outputs a constrained key K_C.

$\mathsf{CPRF.ConstrainEval}(\mathsf{pp}, \mathsf{K}_C, x) :\to y$: The constrained evaluation algorithm takes as input a public parameter pp, a constrained key K_C, and an input $x \in \mathcal{D}$ and outputs $y \in \mathcal{R}$.

Correctness. We define the notion of *correctness* for CPRFs. We say a CPRF Π_{CPRF} is correct if for all $\kappa \in \mathbb{N}$, $C \in \mathcal{C}_\kappa$, and $x \in \mathcal{D}_\kappa$ such that $C(x) = 1$, we have

$$\Pr\left[\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{K}, x) = \mathsf{CPRF.ConstrainEval}(\mathsf{pp}, \mathsf{K}_C, x) \;\middle|\; \begin{array}{c} \mathsf{pp} \xleftarrow{\$} \mathsf{CPRF.Setup}(1^\kappa) \\ \mathsf{K} \xleftarrow{\$} \mathsf{CPRF.Gen}(\mathsf{pp}) \\ \mathsf{K}_C \xleftarrow{\$} \mathsf{CPRF.Constrain}(\mathsf{K}, C) \end{array} \right] \leq \mathsf{negl}(\kappa).$$

Pseudorandomness on Constrained Points. We define the notion of (adaptive) *pseudorandomness on constrained points* for CPRFs. Informally, we require it infeasible to evaluate on a point when only given constrained keys that are constrained on that particular point. For any $C : \mathcal{D} \to \{0,1\}$, let $\mathsf{ConPoint} : \mathcal{C} \to \mathcal{D}$ be a function which outputs the set of all constrained points $\{x \mid C(x) = 0\}$. Here ConPoint is not necessarily required to be efficiently computable.

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

[5] Here, $\mathcal{D}_\kappa$ is not restricted to be a set over binary strings as we also consider arithmetic circuits.

Setup: At the beginning of the game, the challenger prepares the public parameter $\mathsf{pp} \xleftarrow{\$} \mathsf{CPRF.Setup}(1^\kappa)$, master key $\mathsf{K} \xleftarrow{\$} \mathsf{CPRF.Gen}(\mathsf{pp})$ and two sets $S_{\mathsf{eval}}, S_{\mathsf{con}}$ initially set to be empty. It sends pp to $\mathcal{A}$.

Queries: During the game, $\mathcal{A}$ can adaptively make the following two types of queries:

-**Evaluation Queries**: Upon a query $x \in \mathcal{D}$, the challenger evaluates $y \xleftarrow{\$} \mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{K}, x)$ and returns $y \in \mathcal{R}$ to $\mathcal{A}$. It then updates $S_{\mathsf{eval}} \leftarrow S_{\mathsf{eval}} \cup \{x\}$.

-**Constrained Key Queries**: Upon a query $C \in \mathcal{C}$, the challenger runs $\mathsf{K}_C \xleftarrow{\$} \mathsf{CPRF.Constrain}(\mathsf{K}, C)$ and returns K_C to $\mathcal{A}$. It then updates $S_{\mathsf{con}} \leftarrow S_{\mathsf{con}} \cup \{C\}$.

Challenge Phase: At some point, $\mathcal{A}$ chooses its target input $x^* \in \mathcal{D}$ such that $x^* \notin S_{\mathsf{eval}}$ and $x^* \in \mathsf{ConPoint}(C)$ for all $C \in S_{\mathsf{con}}$. The challenger chooses a random bit $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. If $\mathsf{coin} = 0$, it evaluates $y^* \xleftarrow{\$} \mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{K}, x^*)$. If $\mathsf{coin} = 1$, it samples a random value $y^* \xleftarrow{\$} \mathcal{R}$. Finally, it returns y^* to $\mathcal{A}$.

Queries: After the challenge phase, $\mathcal{A}$ may continue to make queries with the added restriction that it cannot query x^* as the evaluation query and cannot query any circuit C such that $C(x^*) = 1$ as the constrained key query.

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 3.14. *A CPRF Π_{CPRF} is said to be (adaptively) pseudorandom on constrained points if for all PPT adversary $\mathcal{A}$, $|\Pr[\mathcal{A}\ wins] - 1/2| = \mathsf{negl}(\kappa)$ holds.*

Remark 3.15 (Selective Security of Constrained Keys). In case all the constrained key queries made by the adversary must be provided before the **Setup** phase, we say it is *selective* pseudorandom on constrained points. All known constructions of CPRFs for non-trivial predicates based on standard assumptions in the standard model satisfy only selective security. Constructions that achieve adaptive security are based on stronger assumptions (e.g. IO, multilinear maps) or are situated in the ROM.

Remark 3.16 (Collusion Resistance). We can adjust the strength of the above notion by imposing a restriction on the number of constrained keys an adversary can query. In case the adversary can query at most one constrained key, it is called *single-key* secure. In case we can tolerate up to Q constrained key queries, we say it is *Q-collusion resistance*. Except for the CPRF for the limited class of prefix-fixing predicates of [BFP+15], similarly with adaptive security above, we require strong assumptions or RO to achieve collusion resistance.

1-key privacy. We adopt the indistinguishability notion of 1-key privacy that was introduced by Boneh et al. [BLW17].[6] This property is sometimes known better as "constraint-hiding". We note that the simulation-based definition of Canetti and Chen [CC17] is stronger, but we are unable to prove security in this setting. Essentially, there is a disparity between the number of constrained queries that we permit, and the number of constraint-hiding keys that we can prove security for.

Let $\mathcal{C}$ denote the class of predicates that are associated to constrained keys.

Setup: At the beginning of the game, the challenger prepares the public parameter $\mathsf{pp} \xleftarrow{\$} \mathsf{CPRF.Setup}(1^\kappa)$ and the master key $\mathsf{K} \xleftarrow{\$} \mathsf{CPRF.Gen}(\mathsf{pp})$, and sends pp to $\mathcal{A}$.

Constrained Key Query: $\mathcal{A}$ specifies two predicate circuits $C_0, C_1 \in \mathcal{C}$. The challenger chooses a random bit $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. The challenger then runs $\mathsf{K}_{\mathsf{coin}} \xleftarrow{\$} \mathsf{CPRF.Constrain}(\mathsf{K}, C_{\mathsf{coin}})$ and returns $\mathsf{K}_{\mathsf{coin}}$ to $\mathcal{A}$.

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

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

[6] Note that the original definition is for m-key privacy but we only consider that $m = 1$ only, as this is relevant to our work.

Definition 3.17. *A CPRF Π_{CPRF} is said to satisfy perfect weak 1-key privacy if for all PPT adversaries $\mathcal{A}$, then $|\Pr[\mathcal{A} \text{ wins}] - 1/2| = 0$ holds.*

Remark 3.18. The version of key privacy that we use above is better known as *weak* key privacy [BLW17]. This is because the adversary has no access to an evaluation oracle. We note that the main applications of PCPRFs are instantiable under weak key privacy. As a result, we do not lose anything by considering the weaker security guarantee.[7] It should also be noted that the previous definitions of key privacy were settled computationally. In this work we actually satisfy the notion of *perfect* key privacy due to the lack of structure in our constrained keys.

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 [GGM86, HILL99]. 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 $F = \{\{x\} \mid x \in \{0,1\}^n\}$ [BW13] or the selectively-secure and collusion-resistant CPRF for prefix-fixing predicates by [BFP$^+$15].

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 Appendix B 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 $C^{\mathsf{BF}}_v : \{0,1\}^\ell \to \{0,1\}$ associated with v as*

$$C^{\mathsf{BF}}_v(x) = \bigwedge_{i=1}^{\ell} \left((v_i \overset{?}{=} x_i) \bigvee (v_i \overset{?}{=} *) \right),$$

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

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

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

We also define a helper function $G^{\mathsf{BF}}_{\mathsf{auth}}$ 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.

[7] There is also no need for an admissibility requirement.

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 $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ be any PRF with input length n and output length k.

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

$\mathsf{CPRF.Gen}(1^\kappa) \to \mathsf{K}$: 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

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

$\mathsf{CPRF.Eval}(\mathsf{K}, x) :\to y$: 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 $y \in \{0,1\}^k$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C^{\mathsf{BF}}_v) :\to \mathsf{K}_v$: 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 $G^{\mathsf{BF}}_{\mathsf{auth}}(v_T) = \{w \in \{0,1\}^Q \mid C^{\mathsf{BF}}_{v_T}(w) = 1\}$.

$\mathsf{CPRF.ConstrainEval}(\mathsf{K}_v, x) :\to y$: On input the constrained key K_v and an input $x \in \{0,1\}^n$, it first parses $\big(\widetilde{\mathsf{K}}_{T,w}\big)_{T\in[n]^Q, w\in\{0,1\}^Q} \leftarrow \mathsf{K}_v$. 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 $y \in \{0,1\}^k$.

4.3 Correctness

We check correctness of our CPRF. Let C^{BF}_v be any bit-fixing predicate in $\mathcal{C}^{\mathsf{BF}}_n$. Put differently, let us fix an arbitrary $v \in \{0,1,*\}^n$. Then, by construction we have

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

Now, for any $x \in \{0,1\}^n$ such that $C^{\mathsf{BF}}_v(x) = 1$, by definition of the bit-fixing predicate, we have

$$\bigwedge_{i=1}^{n} \Big((v_i \overset{?}{=} x_i) \bigvee (v_i \overset{?}{=} *)\Big) = 1.$$

Then, by definition of function $G^{\mathsf{BF}}_{\mathsf{auth}}$, we have $x_T \in G^{\mathsf{BF}}_{\mathsf{auth}}(v_T)$ for any $T \in [n]^Q$ since we have $C^{\mathsf{BF}}_{v_T}(x_T) = 1$ if $C^{\mathsf{BF}}_{v}(x)$ = 1. In particular, we have

$$\widetilde{\mathsf{K}}_{T,x_T} = \mathsf{K}_{T,x_T} \in \mathsf{K}_v \text{ for all } T \in [n]^Q.$$

Therefore, since CPRF.Eval and CPRF.ConstrainEval are computed exactly in the same way, using the same PRF keys, correctness holds.

4.4 Security

Theorem 4.2. *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 $Q = O(1)$.*

Proof of Theorem 4.2. We show the theorem by considering the following sequence of games between an adversary $\mathcal{A}$ against the pseudorandomness on constrained points security game and the challenger. In the following, for simplicity, we say an adversary $\mathcal{A}$ against the CPRF pseudorandomness game. Below, let E_i denote the probability that $\widehat{\mathsf{coin}} = \mathsf{coin}$ holds in Game_i. Recall that $\mathcal{A}$ makes at most Q-constrained key queries, where Q is a constant.

Game_0**:** This is defined as the ordinary CPRF pseudorandomness game played between $\mathcal{A}$ and the challenger. In particular, at the beginning of the game the challenger prepares the empty sets $S_{\mathsf{eval}}, S_{\mathsf{con}}$. In this game, the challenger responds to the queries made by $\mathcal{A}$ as follows:

- When $\mathcal{A}$ submits $x \in \{0,1\}^n$ as the evaluation query, the challenger returns $y \xleftarrow{\$} \mathsf{CPRF.Eval}(\mathsf{K}, x)$ to $\mathcal{A}$ and updates $S_{\mathsf{eval}} \leftarrow S_{\mathsf{eval}} \cup \{x\}$.
- When $\mathcal{A}$ submits $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_n$ as the j^{th} ($j \in [Q]$) constrained key query, the challenger returns $\mathsf{K}_{v^{(j)}} \xleftarrow{\$} \mathsf{CPRF.Constrain}(\mathsf{K}, C^{\mathsf{BF}}_{v^{(j)}})$ to $\mathcal{A}$ and updates $S_{\mathsf{con}} \leftarrow S_{\mathsf{con}} \cup \{C^{\mathsf{BF}}_{v^{(j)}}\}$.

Furthermore, recall that when $\mathcal{A}$ submits the target input $x^* \in \{0,1\}^n$ as the challenge query, we have the restriction $x^* \notin S_{\mathsf{eval}}$ and $x^* \in \mathsf{ConPoint}(C^{\mathsf{BF}}_{v^{(j)}})$ for all $C^{\mathsf{BF}}_{v^{(j)}} \in S_{\mathsf{con}}$. Here, the latter condition is equivalent to

$$\bigwedge_{i=1}^{n} \left((v_i^{(j)} \overset{?}{=} x_i^*) \bigvee (v_i^{(j)} \overset{?}{=} *) \right) = 0 \quad \text{for all} \quad C^{\mathsf{BF}}_{v^{(j)}} \in S_{\mathsf{con}}. \tag{1}$$

By definition, we have $|\Pr[\mathsf{E}_0] - 1/2| = \epsilon$.

Game_1**:** In this game, we add an extra abort condition for the challenger. Specifically, at the end of the game, the challenger samples a random set $T^* \xleftarrow{\$} [n]^Q$. Let us set $T^* = (t_1, \cdots, t_Q)$. The challenger further samples $b^*_{t_j} \xleftarrow{\$} \{0,1\}$ for all $j \in [Q]$. Let $b^*_{T^*} := b_{t_1} b_{t_2} \cdots b_{t_Q} \in \{0,1\}^Q$. Then, the challenger checks whether the following equation holds with respect to the constrained key queries and the challenge query made by the adversary $\mathcal{A}$ at the end of the game:

- The challenger aborts if there exists $j \in [Q]$ such that
$$\left((v^{(j)}_{t_j} \neq b^*_{t_j}) \bigwedge (v^{(j)}_{t_j} \neq *) \right) = 0 \tag{2}$$
is satisfied.
- The challenger aborts if x^* does not satisfy
$$(b^*_{T^*} \overset{?}{=} x^*_{T^*}) = \bigwedge_{j \in [Q]} (b^*_{t_j} \overset{?}{=} x^*_{t_j}) = 1. \tag{3}$$

- The challenger aborts if $(T^*, b^*_{T^*})$ chosen by the challenger does not equal to the *first pair* (with respect to some pre-defined order over $[n]^Q \times \{0,1\}^Q$ such as the lexicographic order) that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). Note that it is possible to efficiently find such a pair by enumerating over $[n]^Q \times \{0,1\}^Q$ since $Q = O(1)$.[8]

When the challenger aborts, it substitutes the guess $\widehat{\mathsf{coin}}$ outputted by $\mathcal{A}$ with a random bit. We call this event abort.

As we will show in Lemma 4.3, there exists at least a single pair $(T^*, b^*_{T^*}) \in [n]^Q \times \{0,1\}^Q$ that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). Therefore, the event abort occurs with probability $1 - 1/(2n)^Q$. Furthermore, it can be seen that abort occurs independently from the view of $\mathcal{A}$. Therefore, we have

$$\begin{aligned}|\Pr[\mathsf{E}_1] - 1/2| &= |\Pr[\mathsf{E}_0]\cdot\Pr[\neg\mathsf{abort}] + (1/2)\cdot\Pr[\mathsf{abort}] - 1/2|\\ &= |\Pr[\mathsf{E}_0]\cdot(1/(2n)^Q) + (1/2)\cdot(1 - 1/(2n)^Q) - 1/2|\\ &= \epsilon/(2n)^Q,\end{aligned}$$

where we used the fact that $\widehat{\mathsf{coin}}$ is randomly chosen and thus equals to coin with probability $1/2$ when abort occurs.

Game_2**:** Recall that in the previous game, the challenger aborts at the end of the game, if the abort condition is satisfied. In this game, we change the game so that the challenger chooses T^* and $b^*_{T^*}$ at the beginning of the game and aborts as soon as either $\mathcal{A}$ makes a constrained key query $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_n$ that does not satisfy Equation (2) or a challenge query for x^* that does not satisfy Equation (3). Furthermore, it aborts if $(T^*, b^*_{T^*})$ is not the first pair that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). Since this is only a conceptual change, we have

$$\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_1].$$

Game_3**:** In this game, we change how the challenger responds to the challenge query when $\mathsf{coin} = 0$. For all the evaluation query and constrained key query, the challenger acts exactly the same way as in the previous game. In the previous game Game_2, when the adversary submits the target input $x^* \in \{0,1\}^n$ as the challenge query, the challenger first checks whether the condition in Equation (3) holds. If not it aborts. Otherwise, it samples $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. In case $\mathsf{coin} = 0$, it computes $\mathsf{CPRF.Eval}(\mathsf{K}, x^*)$ as

$$y^* = \bigoplus_{T \in [n]^Q} \mathsf{PRF.Eval}(\mathsf{K}_{T, x^*_T}, x^*) \tag{4}$$

using the master key

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

that it constructed at the beginning of the game, where $\mathsf{K}_{T,w}, \widehat{\mathsf{K}}_{T,w} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [n]^Q$ and $w \in \{0,1\}^Q$. Due to the condition in Equation (3), i.e., $b^*_{T^*} = x^*_{T^*} \in \{0,1\}^Q$, we can rewrite Equation (4) as

$$y^* = \mathsf{PRF.Eval}(\mathsf{K}_{T^*, b^*_{T^*}}, x^*) \oplus \left(\bigoplus_{T \in [n]^Q \setminus \{T^*\}} \mathsf{PRF.Eval}(\mathsf{K}_{T, x^*_T}, x^*)\right). \tag{5}$$

In this game Game_3, when $\mathsf{coin} = 1$, the challenger instead samples a random $\bar{y}^* \xleftarrow{\$} \{0,1\}^k$ and returns the following to $\mathcal{A}$ instead of returning y^* to $\mathcal{A}$ as in Equation (5):

$$y^* = \bar{y}^* \oplus \left(\bigoplus_{T \in [n]^Q \setminus \{T^*\}} \mathsf{PRF.Eval}(\mathsf{K}_{T, x^*_T}, x^*)\right). \tag{6}$$

[8] One may wonder why the final condition for the abort is necessary, because the reduction in the proof of Lemma 4.4 works even without it. This additional abort step is introduced to make the probability of abort to occur independently of the choice of the constrained key queries and the challenge query made by the adversary. Without this step, we cannot lower bound $|\Pr[\mathsf{E}_1] - 1/2|$. Similar problem was identified by Waters [Wat05], who introduced “the artificial abort step" to resolve it. Our analysis here is much simpler because we can compute the abort probability exactly in our case.

We show in Lemma 4.4 that

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\kappa)$$

assuming pseudorandomness of the underlying PRF Π_{PRF}. In this game Game_3, the distribution of y^* for $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ are exactly the same since $\mathcal{A}$ has not made an evaluation query on x^* and $\mathsf{K}_{T^*, b^*_{T^*}}$ is not given through any of the constrained key query. Concretely, $\bar{y}^*$ is distributed uniform random regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$ and thus the value of coin is information theoretically hidden to $\mathcal{A}$. Therefore, we have

$$\Pr[\mathsf{E}_3] = 1/2.$$

Combining everything together with Lemma 4.3 and Lemma 4.4, we have

$$\epsilon = |\Pr[\mathsf{E}_0] - 1/2| \leq (2n)^Q \cdot (|\Pr[\mathsf{E}_3] - 1/2| + \mathsf{negl}(\kappa)) = \mathsf{negl}(\kappa),$$

where the last equality follows by recalling that $n = \mathsf{poly}(\kappa)$ and Q a constant.

Lemma 4.3. *In* Game_1, *we have*

$$\left\{ (T^*, b^*_{T^*}) \in [n]^Q \times \{0,1\}^Q \;\middle|\; \begin{array}{l} (T^*, b^*_{T^*}) \text{ satisfies Equation (2)} \\ \text{for all } j \in [Q], \text{ and Equation (3)} \end{array} \right\} \neq \emptyset.$$

Proof. By the restriction posed on $\mathcal{A}$ in the game, for all $j \in [Q]$, there exists $t^{(j)} \in [n]$ such that

$$v^{(j)}_{t^{(j)}} = 1 - x^*_{t^{(j)}}.$$

Let us denote $\bar{T} := (t^{(1)}, \cdots, t^{(Q)}) \in [n]^Q$ and $\bar{b}_{\bar{T}} := x^*_{\bar{T}} \in \{0,1\}^Q$. It is easy to check that Equation (2) for all $j \in [Q]$ and Equation (3) hold if $T^* = \bar{T}$ and $b^*_{T^*} = \bar{b}_{\bar{T}}$. □

Lemma 4.4. *We have* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\kappa)$ *assuming that the underlying PRF* Π_{PRF} *satisfies adaptive pseudorandomness.*

Proof. For the sake of contradiction, let us assume an adversary $\mathcal{A}$ that distinguishes Game_2 and Game_3 with non-negligible probability ϵ'. We then construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of Π_{PRF} with the same probability. The adversary $\mathcal{B}$ proceeds as follows.

At the beginning of the game $\mathcal{B}$ samples a random tuple $T^* = (t_1, \cdots, t_Q) \xleftarrow{\$} [n]^Q$ and $b^*_{t_j} \xleftarrow{\$} \{0,1\}$ for all $j \in [Q]$ as in the Game_2-challenger. Let $b^*_{T^*} := b_{t_1} b_{t_2} \cdots b_{t_Q} \in \{0,1\}^Q$. Then, it further samples $\mathsf{K}_{T,w} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [n]^Q$ and $w \in \{0,1\}^Q$ *except* for $(T^*, b^*_{T^*})$. It then sets the (simulated) master key K^* as

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

Here, $\mathcal{B}$ implicitly sets $\mathsf{K}_{T^*, b^*_{T^*}}$ as the PRF key used by its PRF challenger. Finally, $\mathcal{B}$ prepares two empty sets $S_{\mathsf{eval}}, S_{\mathsf{con}}$. $\mathcal{B}$ then simulates the response to the queries made by $\mathcal{A}$ as follows:

- When $\mathcal{A}$ submits $x \in \{0,1\}^n$ as the evaluation query, $\mathcal{B}$ checks whether $x_{T^*} = b^*_{T^*}$. If not, then it can use the simulated master key K^* to compute

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

  Otherwise, it makes an evaluation query to its PRF challenger on the input x. When it receives back $\bar{y}$ from the PRF challenger, $\mathcal{B}$ computes the output as

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

  Finally, $\mathcal{B}$ returns y to $\mathcal{A}$ and updates $S_{\mathsf{eval}} \leftarrow S_{\mathsf{eval}} \cup \{x\}$. Note that by the specification of the PRF challenger, we have $\bar{y} = \mathsf{PRF.Eval}(\mathsf{K}_{T^*, b^*_{T^*}}, x)$.

- When $\mathcal{A}$ submits $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_n$ as the j^{th} ($j \in [Q]$) constrained key query, $\mathcal{B}$ checks whether the condition in Equation (2) holds. If not it aborts and outputs a random bit. Otherwise, it returns the following constrained key $\mathsf{K}_{v^{(j)}}$ to $\mathcal{A}$:

$$\mathsf{K}_{v^j} = \left(\widetilde{\mathsf{K}}_{T,w}\right)_{T\in[n]^Q, w\in\{0,1\}^Q},$$

  where $\widetilde{\mathsf{K}}_{T,w} = \mathsf{K}_{T,w}$ if and only if $w \in G^{\mathsf{BF}}_{\mathsf{auth}}(v^{(j)}_T) = \{w \in \{0,1\}^Q \mid C^{\mathsf{BF}}_{v^{(j)}_T}(w) = 1\}$ and $\widetilde{\mathsf{K}}_{T,w} = \widehat{\mathsf{K}}_{T,w}$ otherwise. Here, $\mathcal{B}$ can prepare all the PRF keys since the condition in Equation (2) guarantees us that we have $b^*_{T^*} \notin G^{\mathsf{BF}}_{\mathsf{auth}}(v^{(j)}_{T^*})$, or equivalently, $C^{\mathsf{BF}}_{v^{(j)}_{T^*}}(b^*_{T^*}) = 0$. Namely, $\widetilde{\mathsf{K}}_{T^*,b^*_{T^*}} = \widehat{\mathsf{K}}_{T^*,b^*_{T^*}}$ and so $\mathsf{K}_{T^*,b^*_{T^*}}$ is not included in $\mathsf{K}_{v^{(j)}}$.

- When $\mathcal{A}$ submits the target input $x^* \in \{0,1\}^n$ as the challenge query, $\mathcal{B}$ checks whether the condition in Equation (3) holds. If not it aborts and outputs a random bit. Otherwise, $\mathcal{B}$ queries its PRF challenger on x^* as its challenge query and receives back $\bar{y}^*$. It then computes y^* as in Equation (6) and returns y^* to $\mathcal{A}$. Here, since Equation (3) holds, $\mathsf{K}_{T^*,b^*_{T^*}}$ must be required to compute on input x^*.

Finally, $\mathcal{A}$ outputs its guess $\widehat{\mathsf{coin}}$. $\mathcal{B}$ then checks whether $(T^*, b^*_{T^*})$ is the first pair that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). If it does not hold, $\mathcal{B}$ outputs a random bit. Otherwise, $\mathcal{B}$ outputs $\widehat{\mathsf{coin}}$ as its guess.

This completes the description of $\mathcal{B}$. It is easy to check that in case $\mathsf{coin} = 0$, $\mathcal{B}$ receives $\bar{y}^* \xleftarrow{\$} \mathsf{PRF.Eval}(\mathsf{K}_{T^*,b^*_{T^*}}, x^*)$, hence $\mathcal{B}$ simulates Game_2 perfectly. Otherwise in case $\mathsf{coin} = 1$, $\mathcal{B}$ receives $\bar{y}^* \xleftarrow{\$} \{0,1\}^k$, hence $\mathcal{B}$ simulates Game_3 perfectly. Therefore, we conclude that $\mathcal{B}$ wins the PRF pseudorandomness game with probability exactly ϵ'. Assuming that Π_{PRF} is pseudorandom, this is a contradiction, hence, ϵ' must be negligible. □

This completes the proof. □

Theorem 4.5. *The above CPRF Π_{CPRF} for the bit-fixing predicate $\mathcal{C}^{\mathsf{BF}}$ satisfies perfect weak 1-key privacy.*

Proof. Notice that the master key is of the form:

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

where $\mathsf{K}_{T,w}, \widehat{\mathsf{K}}_{T,w} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$. Let $v^{(0)}, v^{(1)} \in \{0,1,*\}^n$ be the two bit-fixing strings that the adversary $\mathcal{A}$ queries. Then, $\mathcal{A}$ receives either one of the following two distributions:

- $\left(\widetilde{\mathsf{K}}^{(0)}_{T,w}\right)_{T\in[n]^Q, w\in\{0,1\}^Q}$ where $\widetilde{\mathsf{K}}^{(0)}_{T,w} = \mathsf{K}_{T,w}$ if and only if $w \in G^{\mathsf{BF}}_{\mathsf{auth}}(v^{(0)}_T)$, and $\widetilde{\mathsf{K}}_{T,w} = \widehat{\mathsf{K}}_{T,w}$ otherwise.
- $\left(\widetilde{\mathsf{K}}^{(1)}_{T,w}\right)_{T\in[n]^Q, w\in\{0,1\}^Q}$ where $\widetilde{\mathsf{K}}^{(1)}_{T,w} = \mathsf{K}_{T,w}$ if and only if $w \in G^{\mathsf{BF}}_{\mathsf{auth}}(v^{(1)}_T)$, and $\widetilde{\mathsf{K}}_{T,w} = \widehat{\mathsf{K}}_{T,w}$ otherwise.

Notice that both the distributions are made up entirely of keys sampled from $\mathsf{PRF.Gen}$. Moreover, $\mathcal{A}$ cannot compare outputs under the constrained key and the real master key since $\mathcal{A}$ has no access to the evaluation oracle in this setting. Therefore, the two distributions are perfectly indistinguishable and the proof of weak key privacy is complete. □

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 $C_{\mathbf{v}}(\mathbf{w}) = (\langle \mathbf{v}, \mathbf{w}\rangle \overset{?}{=} 0)$, 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 3.7.
- 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 $n, m, q, p, \beta, \bar{\beta}$ 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 $\mathbb{Z}_q^{n\times\ell}$, respectively.

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

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

$\mathsf{CPRF.Gen}(\mathsf{pp})$: 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

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

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

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

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

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

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.

- $m > (n+1)\log q + \omega(\log n)$ (For Lemma 5.4)
- $\alpha_{\mathsf{LWE}} q > 2\sqrt{h}$ (For Lemma 5.5)
- $\|\mathbf{x}^\top \mathbf{S}^\top \mathbf{E} \mathbf{R}_x\|_\infty \leq \alpha_1$ for all $\mathbf{x} \in \mathcal{D}$ (For Lemma 5.6)
- $\|\ell B \mathbf{d}^\top \mathbf{E} \mathbf{R}_x\|_\infty \leq \alpha_2$ for all $\mathbf{x} \in \mathcal{D}$ (For Lemma 5.6)
- $q = 2mpB^\ell \cdot (\alpha_1 + \alpha_2) \cdot \kappa^{\omega(1)}$ for all $\mathbf{x} \in \mathcal{D}$ (For Lemma 5.6)
- $\bar{\beta} \geq \beta B$ and $\bar{\beta} = n\ell\beta B \cdot \kappa^{\omega(1)}$ (For Lemma 5.7)
- $\bar{n} := n \cdot \log(2\beta + 1) - h \cdot \log q = \Omega(\kappa)$ (For Lemma 5.8)

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 Section 3.2). 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:

$$\begin{array}{ccc} n = \ell h^{1.1}, & m = \ell^2 h^{1.2}, & q = \ell^{15.5} h^{9.5} B^{\ell + 2.5} \kappa^{2\log\kappa}, \\ p = 2, & \alpha_{\mathsf{LWE}} = 2\sqrt{h} \cdot q^{-1}, & \zeta = \bar{n} + \ell \cdot \log B, \\ \alpha_1 = \alpha_2 = \ell^{13.5} h^{8.3} \kappa^{\log\kappa}, & \beta = 1, & \bar{\beta} = \ell^2 h^{1.1} B \kappa^{\log\kappa}. \end{array}$$

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 Lemma 3.11. 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, assuming hardness of the $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha_{\mathsf{LWE}}q}}$ problem.*

Proof of Theorem 5.1. Let $\mathcal{A}$ be a PPT adversary that wins the CPRF pseudorandomness game. In addition, let $\epsilon = \epsilon(\kappa)$ and $Q = Q(\kappa)$ be its advantage and the upper bound on the number of evaluation queries, respectively. By assumption, $Q(\kappa)$ is polynomially bounded and there exists a noticeable function $\delta(\kappa)$ such that $\epsilon(\kappa) \geq \delta(\kappa)$ holds for infinitely many $\kappa \in \mathbb{N}$. By the property of the balanced admissible hash (see Definition 3.6, Item 1), we have $\Pr[K \xleftarrow{\$} \mathsf{PrtSmp}(1^\kappa, Q(\kappa), \delta(\kappa)) : K \in \{0,1,\perp\}^\ell] = 1$ for all sufficiently large κ. Therefore, in the following, we assume that this condition always holds.

We show security of the scheme via the following games. We will bound the differences in $\mathcal{A}$'s view of each adjacent games and prove that no (even computationally unbounded) $\mathcal{A}$ has any winning advantage in the final game.

Game_0 : This is the real security game where the adversary $\mathcal{A}$ outputs $\widehat{\mathsf{coin}} \in \{0,1\}$. Note that $\mathcal{A}$ queries the constrained key query at most once.

Game_1 : In this game, we change Game_0 so that the challenger runs an additional procedure at the beginning of the game and performs an additional check at the end of the game. At the beginning of the game, the challenger samples $K \leftarrow \mathsf{PrtSmp}(1^\kappa, Q, \delta)$. After the adversary $\mathcal{A}$ outputs its guess $\widehat{\mathsf{coin}}$, the challenger checks whether the following condition is met:

$$P_K(\mathsf{H}_{\mathsf{adm}}(\mathbf{x}_1)) = 1 \wedge \cdots \wedge P_K(\mathsf{H}_{\mathsf{adm}}(\mathbf{x}_Q)) = 1 \wedge P_K(\mathsf{H}_{\mathsf{adm}}(\mathbf{x}^*)) = 0, \tag{7}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_Q \in \mathcal{D}$ are inputs made by $\mathcal{A}$ to the evaluation query and $\mathbf{x}^*$ is the target input chosen at the challenge phase. If it does not hold, the challenger ignores the bit $\widehat{\mathsf{coin}}$ output by $\mathcal{A}$ and replaces it with a fresh random coin $\widehat{\mathsf{coin}} \xleftarrow{\$} \{0,1\}$. In this case, we say the game *aborts*. We note that K, which is sampled at the beginning of the game, is not used until the end of the game.

Game_2 : In this game, we change how the challenger generates the matrix $\mathbf{A}$ in the public parameters pp. At the beginning of the game the challenger samples random matrices $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{R} \xleftarrow{\$} \{-1,0,1\}^{m \times mu}$. It further samples $K \leftarrow \mathsf{PrtSmp}(1^\kappa, Q, \delta)$ and computes $\mu = \mathsf{Encode}(K)$ where $\mu \in \{0,1\}^u$ (see Lemma 3.8). Finally, the challenger sets $\mathbf{A} = \mathbf{A}_0\mathbf{R} + \mu \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times mu}$. Otherwise the challenger is defined identically to the previous Game_1.

Game_3 : Recall that in Game_2, the challenger checked whether the non-abort condition (Equation (7)) holds or not at the end of the game. In this game, we modify the challenger so that it forces $\mathcal{A}$ to output a random bit $\widehat{\mathsf{coin}}$ as soon as the non-abort condition becomes false, i.e., if Equation (7) does not hold.

Game_4 : In this game, we change how the challenger generates the matrix $\mathbf{A}_0$. The only difference between the previous game is that the challenger samples $\mathbf{A}_0$ as

$$\mathbf{A}_0 = \bar{\mathbf{A}}\mathbf{F} + \mathbf{E} \in \mathbb{Z}_q^{n \times m}, \tag{8}$$

where $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times h}$, $\mathbf{F} \leftarrow \mathbb{Z}_q^{h \times m}$, and $\mathbf{E} \leftarrow D_{\mathbb{Z}^{n \times m}, \alpha_{\mathsf{LWE}} q}$.

Game_5 : In this game, we change the challenger to further run an additional check at the end of the game. If the non-abort condition (Equation (7)) is met at the end of the game, the challenger further checks for all $\mathbf{x} \in \{\mathbf{x}_1, \ldots, \mathbf{x}_Q, \mathbf{x}^*\}$ that the following condition holds.

$$\left\lfloor (\mathbf{S}\mathbf{x})^\top \mathbf{A}_x \right\rceil_p = \left\lfloor (\mathbf{S}\mathbf{x})^\top \mathbf{A}_x + \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{E}\mathbf{R}_x \right\rceil_p \tag{9}$$

where $\mathbf{y}^*$ is the vector corresponding to the constraint $C_{\mathbf{y}^*}$ that $\mathcal{A}$ queries as the (single) challenge constrained key query, $\mathbf{A}_x = \mathsf{PubEval}(x, \mathbf{A})$, and $\mathbf{R}_x = \mathsf{TrapEval}(\mu, x, \mathbf{A}, \mathbf{R})$. Here, note that the l.h.s. corresponds to what $\mathcal{A}$ receives for its evaluation query and challenge query when $\mathsf{coin} = 0$. We denote BAD_1 as the event that there exists some $\mathbf{x}$ such that the above condition does not hold. In case event BAD_1 occurs, the challenger ignores the bit $\widehat{\mathsf{coin}}$ output by $\mathcal{A}$ and replaces it with a fresh random coin $\widehat{\mathsf{coin}} \xleftarrow{\$} \{0,1\}$.

Game_6 : In this game, we add one last additional check at the end of the game for the challenger to run. If the non-abort condition (Equation (7)) is met and event BAD_1 (Equation (9)) does not occur, the challenger further checks whether the constrained key $\mathsf{K}_{\mathbf{y}^*} = \mathbf{S}_{\mathbf{y}^*} \in \mathbb{Z}^{n \times \ell}$ satisfies

$$\|\mathbf{S}_{\mathbf{y}^*}\|_\infty \le \bar{\beta} - \beta \cdot B. \tag{10}$$

Here, we denote BAD_2 as the event that the above condition does not hold. In case event BAD_2 occurs, the challenger ignores the bit $\widehat{\mathsf{coin}}$ output by $\mathcal{A}$ and replaces it with a fresh random coin $\widehat{\mathsf{coin}} \xleftarrow{\$} \{0,1\}$.

Game_7 : In this game, we change the challenger so that it outputs to $\mathcal{A}$ a random challenge $\mathbf{v} \leftarrow \mathbb{Z}_p^k$ when $\mathsf{coin} = 1$.

Let E_i denote the event that Game_i returns 1 (i.e., $\widehat{\mathsf{coin}} = \mathsf{coin}$). We prove the following lemmas.

Lemma 5.2. *If* $\mathsf{H}_{\mathsf{adm}}$ *is a balanced admissible hash function and* $|\Pr[\mathsf{E}_0] - 1/2|$ *is non-negligible, then* $|\Pr[\mathsf{E}_1] - 1/2|$ *is non-negligible.*

Proof. We apply Lemma 3.10 to bound $|\Pr[\mathsf{E}_1] - 1/2|$. To apply the lemma, we set the input of $\mathcal{D}$ and $\mathcal{D}'$ to be coin used in the two games, respectively. The output of $\mathcal{D}(\mathsf{coin})$ and $\mathcal{D}'(\mathsf{coin})$ is $(X = (\mathbf{x}_1, \cdots, \mathbf{x}_Q, \mathbf{x}^*), \widehat{\mathsf{coin}})$ in Game_0 and Game_1, respectively. In addition, let $\gamma(X)$ be the probability that Equation (7) holds for $X = (\mathbf{x}_1, \cdots, \mathbf{x}_Q, \mathbf{x}^*)$

where the probability is taken over the randomness used for sampling $K \leftarrow \mathsf{PrtSmp}(1^\kappa, Q, \delta)$. Then, by Lemma 3.10, the following holds for infinitely many $\kappa \in \mathbb{N}$ where $\gamma_{\min} = \min_X \gamma(X)$ and $\gamma_{\max} = \max_X \gamma(X)$:

$$|\Pr[\mathsf{E}_1] - 1/2| \geq \gamma_{\min} \cdot \epsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2} \geq \gamma_{\min} \cdot \delta - \frac{\gamma_{\max} - \gamma_{\min}}{2}$$

Finally, using the property of the balanced admissible hash function (see Definition 3.6, Item 2), the r.h.s. is a noticeable function. Therefore, this implies that $|\Pr[\mathsf{E}_1] - 1/2|$ is noticeable for infinitely many $\kappa \in \mathbb{N}$, and hence, non-negligible. □

Lemma 5.3. *We have* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$ *due to the Leftover Hash Lemma.*

Proof. The claim follows readily from a standard argument using the Leftover Hash Lemma (Lemma 3.4). □

Lemma 5.4. *We have* $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$.

Proof. Since the change is only conceptual, the lemma trivially follows. □

Lemma 5.5. *We have* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| = \mathsf{negl}(\kappa)$ *assuming the hardness of the* $\mathsf{LWE}_{h,m,q,D_{\mathbb{Z},\alpha_{\mathsf{LWE}}q}}$ *problem.*

Proof. The proof is an immediate consequence of a hybrid argument. By viewing each row of $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times h}$ as the secret vector, $\mathbf{F} \in \mathbb{Z}_q^{h \times m}$ as the public matrix, and each row of $\mathbf{E} \in \mathbb{Z}^{n \times m}$ as the secret noise vector, we change each row of $\mathbf{A}_0$ into a random vector in $\mathbb{Z}_q^m$. In particular, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq n \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LWE}_{h,m,q,\chi}}$, where $\chi = D_{\mathbb{Z},\alpha_{\mathsf{LWE}}q}$ and $\mathcal{B}$ is a PPT algorithm against the LWE problem that simulates the view to $\mathcal{A}$ in the obvious way. Therefore, assuming hardness of the LWE problem, we obtained the desired statement. □

Lemma 5.6. *We have* $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| \leq \Pr[\mathsf{BAD}_1] = \mathsf{negl}(\kappa)$.

Proof. Unless event BAD_1 does not occur, Game_4 and Game_5 are identical. Hence, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| \leq \Pr[\mathsf{BAD}_1]$. In the following, we upper bound $\Pr[\mathsf{BAD}_1]$. In fact, we show a stronger result in that Equation (9) does not hold for *all* $\mathbf{x} \in D$ with overwhelming probability. First, we rewrite the left hand side of Equation (9) as follows.

$$\begin{aligned}
(\mathbf{Sx})^\top \mathbf{A}_x &= (\mathbf{Sx})^\top (\mathbf{A}_0 \mathbf{R}_x + \underbrace{(1 - P_K(\mathsf{H}_{\mathsf{adm}}(\mathbf{x})))}_{=: b_{\mathbf{x}} \in \{0,1\}} \cdot \mathbf{G}) \\
&= (\mathbf{Sx})^\top (\bar{\mathbf{A}}\mathbf{F} + \mathbf{E})\mathbf{R}_x + b_{\mathbf{x}} \cdot (\mathbf{Sx})^\top \mathbf{G} \\
&= \mathbf{x}^\top \mathbf{S}^\top \bar{\mathbf{A}}\mathbf{F}\mathbf{R}_x + \mathbf{x}^\top \mathbf{S}^\top \mathbf{E}\mathbf{R}_x + b_{\mathbf{x}} \cdot \mathbf{x}^\top \mathbf{S}^\top \mathbf{G} \\
&= (\underbrace{\mathbf{x}^\top \mathbf{S}^\top \bar{\mathbf{A}}}_{=: \mathbf{v}_{\mathbf{x}}^\top})(\underbrace{\mathbf{F}\mathbf{R}_0}_{=: \mathbf{B}}) + \underbrace{\mathbf{x}^\top \mathbf{S}^\top \bar{\mathbf{A}}\mathbf{F}\mathbf{R}'_x + b_{\mathbf{x}} \cdot \mathbf{x}^\top \mathbf{S}^\top \mathbf{G}}_{=: \mathbf{v}_{\mathbf{x},1}^\top} + \underbrace{\mathbf{x}^\top \mathbf{S}^\top \mathbf{E}\mathbf{R}_x}_{=: \mathbf{v}_{\mathbf{x},2}^\top},
\end{aligned} \tag{11}$$

where the first equality follows from Lemma 3.8, Item 4 and the non-abort condition (Equation (7)), and the last equality follows from Lemma 3.8, Item 5.

We now fix $\mathbf{x} \in \mathcal{D} \backslash \{\mathbf{0}\} := [-B, B]^\ell \backslash \{\mathbf{0}\}$ and compute the probability such that Equation (9) holds. Looking ahead, in the end, we take the union bound on all possible $\mathbf{x}$ and bound the probability of event BAD_1 happening. Here, note that excluding $\mathbf{x} = \mathbf{0}$ is done without loss of generality as it never triggers event BAD_1. Now, first of all, observe that $\mathbf{B}$ in Equation (11) is distributed statistically close to $U(\mathbb{Z}_q^{h \times m})$ since $\mathbf{R}_0 \xleftarrow{\$} \{-1, 0, 1\}^{m \times m}$ conditioned on $\mathbf{F} \xleftarrow{\$} \mathbb{Z}_q^{h \times m}$. In addition, taking randomness over $\mathbf{S} \xleftarrow{\$} \mathbb{Z}^{n \times \ell}$ and $\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}^{n \times h}$, we have $\Pr[\mathbf{v}_{\mathbf{x}} = \mathbf{0} \mod q] \leq q^h$ for any $\mathbf{x} \in \mathcal{D} \backslash \{\mathbf{0}\}$. Then, using Lemma 3.2, we have that $\mathbf{v}_{\mathbf{x}}^\top \mathbf{B}$ is distributed statistically close to $U(\mathbb{Z}_q^m)$. Moreover, we can see that $\mathbf{v}_{\mathbf{x}}^\top \mathbf{B} + \mathbf{v}_{\mathbf{x},1}^\top$ is distributed statistically close to $U(\mathbb{Z}_q^m)$ as well by noticing that $\mathbf{R}_0$ is fresh randomness independent from $\mathbf{R}'_x$ (see Lemma 3.8, Item 5). Therefore, for any $\mathbf{e}_1 \in [-\alpha_1, \alpha_1]^m$ and $\mathbf{e}_2 \in [-\alpha_2, \alpha_2]^m$, due to Lemma 3.3, we have

$$\Pr\left[\left\lfloor \mathbf{v}_{\mathbf{x}}^\top \mathbf{B} + \mathbf{v}_{\mathbf{x},1}^\top + \mathbf{e}_1 \right\rfloor_p \neq \left\lfloor \mathbf{v}_{\mathbf{x}}^\top \mathbf{B} + \mathbf{v}_{\mathbf{x},1}^\top + \mathbf{e}_2 \right\rfloor_p \right] \leq m \cdot \frac{2(\alpha_1 + \alpha_2)p}{q}.$$

By taking the union bound over all possible $\mathbf{x} \in \mathcal{D} \backslash \{\mathbf{0}\}$, we have

$$\Pr\left[\forall \mathbf{x} \in \mathcal{D} \backslash \{\mathbf{0}\},\ \left\lfloor \mathbf{v}_{\mathbf{x}}^\top \mathbf{B} + \mathbf{v}_{\mathbf{x},1}^\top + \mathbf{e}_1 \right\rfloor_p \neq \left\lfloor \mathbf{v}_{\mathbf{x}}^\top \mathbf{B} + \mathbf{v}_{\mathbf{x},1}^\top + \mathbf{e}_2 \right\rfloor_p \right] \leq B^\ell \cdot m \cdot \frac{2(\alpha_1 + \alpha_2)p}{q}. \tag{12}$$

Finally, since with overwhelming probability we have $\|\mathbf{v}_{\mathbf{x},2}\|_\infty = \|\mathbf{x}^\top \mathbf{S}^\top \mathbf{E} \mathbf{R}_x\|_\infty \leq \alpha_1$ and $\|\langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{E} \mathbf{R}_x\|_\infty \leq \alpha_2$ for all $\mathbf{x} \in \mathcal{D}$ owing to our parameter selection, the above Equation (12) implies that Equation (9) does not occur with all but negligible probability as desired. Specifically, we have

$$\Pr[\mathsf{BAD}_1] \leq B^\ell \cdot m \cdot \frac{2(\alpha_1 + \alpha_2)p}{q} + \mathsf{negl}(\kappa) = \mathsf{negl}(\kappa), \tag{13}$$

where the r.h.s follows from our choice of parameters. □

Lemma 5.7. *We have* $|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]| \leq \Pr[\mathsf{BAD}_2] = \mathsf{negl}(\kappa)$.

Proof. Unless event BAD_2 does not occur, Game_5 and Game_6 are identical. Hence, we have $|\Pr[\mathsf{E}_5] - \Pr[\mathsf{E}_6]| \leq \Pr[\mathsf{BAD}_2]$. We compute the probability that event BAD_2 occurs. Consider a random variable m sampled uniformly over $[-\bar{\beta}, \bar{\beta}]$. Then, for any $a \in [-\beta \cdot B, \beta \cdot B]$, we have

$$\begin{aligned}
\Pr_{m \xleftarrow{\$} [-\bar{\beta}, \bar{\beta}]} \left[\ |m + a| \leq \bar{\beta} - \beta \cdot B\ \right] &= \Pr_{m \xleftarrow{\$} [-\bar{\beta}, \bar{\beta}]} \left[-(\bar{\beta} - \beta \cdot B) - a \leq m \leq \bar{\beta} - \beta \cdot B - a \right] \\
&= \frac{2(\bar{\beta} - \beta \cdot B) + 1}{2\bar{\beta} + 1} \\
&\geq 1 - \frac{\beta \cdot B}{\bar{\beta}}.
\end{aligned} \tag{14}$$

Here, note that we implicit rely on the fact that $\bar{\beta} \geq \beta \cdot B$. Next, recall that $\mathsf{K}_{\mathbf{y}} = \mathbf{S} + \mathbf{d} \otimes \mathbf{y}^\top$ where $\mathbf{S} \leftarrow [\bar{\beta}, \bar{\beta}]^{n \times \ell}$ and $\|\mathbf{d} \otimes \mathbf{y}^\top\|_\infty \leq \beta \cdot B$. By fixing on $\mathbf{D} = \mathbf{d} \otimes \mathbf{y}^\top$, we get the following

$$\begin{aligned}
&\Pr_{\mathbf{S} \xleftarrow{\$} [-\bar{\beta}, \bar{\beta}]^{n \times \ell}} [\|\mathbf{S} + \mathbf{D}\|_\infty \leq \bar{\beta} - \beta \cdot B] \\
=& 1 - \Pr_{\mathbf{S} \xleftarrow{\$} [-\bar{\beta}, \bar{\beta}]^{n \times \ell}} [\exists (i,j) \in [n] \times [\ell],\ |\mathbf{S}_{i,j} + \mathbf{D}_{i,j}| > \bar{\beta} - \beta \cdot B] \\
\geq& 1 - \sum_{(i,j) \in [n] \times [\ell]} \Pr_{\mathbf{S}_{i,j} \xleftarrow{\$} [-\bar{\beta}, \bar{\beta}]} [|\mathbf{S}_{i,j} + \mathbf{D}_{i,j}| > \bar{\beta} - \beta \cdot B] \\
\geq& 1 - \frac{n\ell\beta B}{\bar{\beta}},
\end{aligned}$$

where the third inequality follows from the union bound and the final inequality follows from Equation (14). Since this holds for any $\mathbf{D}$ such that $\|\mathbf{D}\|_\infty \leq \beta \cdot B$, we conclude

$$\Pr[\mathsf{BAD}_2] \leq \frac{n\ell\beta B}{\bar{\beta}} = \mathsf{negl}(\kappa), \tag{15}$$

where the r.h.s follows from our choice of parameters. □

Lemma 5.8. *We have* $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_7]| = \mathsf{negl}(\kappa)$ *due to the deterministic randomness extractor.*

Proof. We first show that $\mathbf{d}$ used to create the constrained key $\mathsf{K}_{\mathbf{y}}$ retains sufficient entropy from the view of the adversary $\mathcal{A}$ conditioned on the non-abort condition holding and event BAD_1 and BAD_2 not triggering. Then we use the deterministic randomness extractor, that is, the ζ-wise independent hash function $\mathsf{H}_{\mathsf{wise}}$, to argue that the challenge $\mathsf{H}_{\mathsf{wise}}(\mathbf{z}^*)$ is distributed uniformly random from the view of $\mathcal{A}$ where $\mathbf{z}^* = \lfloor \mathbf{s}^\top \mathbf{A}_{x^*} \rfloor_p$. Note that in case if any of the above condition does not hold, then the advantage of $\mathcal{A}$ is exactly $1/2$.

Let $\mathbf{y}^*$ be the vector corresponding to the constraint $C_{\mathbf{y}^*}$ that $\mathcal{A}$ queries as the (single) challenge constrained key query, and let $S = \{\mathbf{x}_1, \cdots, \mathbf{x}_Q, \mathbf{x}^*\}$ be the set of inputs which $\mathcal{A}$ makes an evaluation query and a challenge query. Due to the modification made in Game_4, if the non-abort condition (Equation (7)) is satisfied, then for every $\mathbf{x} \in S$ the adversary obtains the following response from the challenger.

$$\begin{aligned}
\mathbf{z} &= \left\lfloor \mathbf{s}^\top \mathbf{A}_x \right\rfloor_p \\
&= \left\lfloor \mathbf{s}^\top \mathbf{A}_x + \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{E} \mathbf{R}_x \right\rfloor_p \\
&= \left\lfloor \left(\mathbf{s}_{\mathbf{y}^*} - \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d} \right)^\top \mathbf{A}_x + \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{E} \mathbf{R}_x \right\rfloor_p \\
&= \left\lfloor \mathbf{s}_{\mathbf{y}^*}^\top \mathbf{A}_x - \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \left(\mathbf{A}_0 \mathbf{R}_x + (1 - P_K(\mathsf{H}_{\mathsf{adm}}(\mathbf{x}))) \cdot \mathbf{G} \right) + \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{E} \mathbf{R}_x \right\rfloor_p \\
&= \left\lfloor \mathbf{x}^\top \mathbf{S}_{\mathbf{y}^*}^\top \mathbf{A}_x - \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \bar{\mathbf{A}} \mathbf{F} \mathbf{R}_x + (1 - P_K(\mathsf{H}_{\mathsf{adm}}(\mathbf{x}))) \cdot \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} \right\rfloor_p \\
&= \begin{cases} \left\lfloor \mathbf{x}^\top \mathbf{S}_{\mathbf{y}^*}^\top \mathbf{A}_x - \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \bar{\mathbf{A}} \mathbf{F} \mathbf{R}_x + \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} \right\rfloor_p & \text{if } \mathbf{x} = \mathbf{x}^* \\ \left\lfloor \mathbf{x}^\top \mathbf{S}_{\mathbf{y}^*}^\top \mathbf{A}_x - \langle \mathbf{x}, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \bar{\mathbf{A}} \mathbf{F} \mathbf{R}_x \right\rfloor_p & \text{otherwise} \end{cases}, \qquad (16)
\end{aligned}$$

where the second equality follows from event BAD_1 not occurring; the third equality follows from the definition of $\mathbf{s}_{\mathbf{y}^*}$; the fifth equality follows from $\mathbf{A}_0 = \bar{\mathbf{A}}\mathbf{F} + \mathbf{E}$; and the last follows from the non-abort condition being satisfied. Then, for any $\mathbf{x} \neq \mathbf{x}^*$, it can be observed that $\mathbf{z}$ is completely determined by the values $\mathbf{x}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, \mathbf{d}^\top \bar{\mathbf{A}}, \mathbf{F}, \mathbf{R}$. In other words, since $\mathbf{d}^\top \bar{\mathbf{A}} \in \mathbb{Z}_q^h$, every evaluation query made by $\mathcal{A}$ always reveal the same $h \cdot \log q$ bits of information about $\mathbf{d} \in \mathbb{Z}^n$. On the other hand, in case $\mathbf{x} = \mathbf{x}^*$, we can further rewrite Equation (16) as follows:

$$\mathbf{z}^* = \underbrace{\left\lfloor \mathbf{x}^{*\top} \mathbf{S}_{\mathbf{y}^*}^\top \mathbf{A}_{x^*} - \langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \bar{\mathbf{A}} \mathbf{F} \mathbf{R}_{x^*} \right\rfloor_p}_{=:\mathbf{v}^*} + \left\lfloor \langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} \right\rfloor_p + \mathbf{e}_{\mathbf{d},\mathbf{x}^*}^\top, \qquad (17)$$

for some $\mathbf{e}_{\mathbf{d},\mathbf{x}^*} \in \{-1, 0, 1\}^m$. We now argue that given $\mathbf{z}^*$ and $\mathbf{v}^*$ and conditioned on $\langle \mathbf{x}^*, \mathbf{y}^* \rangle \neq 0$, $\mathbf{d}$ is uniquely defined. Indeed, we have

$$\lfloor \langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} \rfloor_p + \mathbf{e}_{\mathbf{d},\mathbf{x}^*} = (p/q) \cdot \langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} + \mathbf{e}_{\mathbf{d},\mathbf{x}^*}^\top + \mathbf{t}_{\mathbf{d},\mathbf{x}^*}^\top$$

for some $\mathbf{t}_{\mathbf{d},\mathbf{x}^*} \in (-1, 1)^m$, and hence,

$$\begin{aligned}
\langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} + \mathbf{e}'_{\mathbf{d},\mathbf{x}^*} &= (q/p) \cdot \left(\lfloor \langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}^\top \mathbf{G} \rfloor_p + \mathbf{e}_{\mathbf{d},\mathbf{x}^*}^\top \right) \\
&= (q/p) \cdot (\mathbf{z}^* - \mathbf{v}^*)
\end{aligned}$$

for some $\mathbf{e}'_{\mathbf{d},\mathbf{x}^*} \in (-2 \cdot q/p,\ 2 \cdot q/p)^m$, where the last equality follows from Equation (17). Finally, using standard facts on inverting LWE samples with $\mathbf{G}$ (see [MP12], Section 4 for example), as long as $2 \cdot (q/p) \leq q/4$, the value $\langle \mathbf{x}^*, \mathbf{y}^* \rangle \cdot \mathbf{d}$ is uniquely defined over $\mathbb{Z}_q^n$. Since $\langle \mathbf{x}^*, \mathbf{y}^* \rangle$, $\|\mathbf{d}\|_\infty$ are much smaller than $\sqrt{q}$ and $\langle \mathbf{x}^*, \mathbf{y}^* \rangle \neq 0$, we conclude that $\mathbf{d}$ is unique over $\mathbb{Z}^n$ as well.

We are now ready to argue that conditioned on the view of $\mathcal{A}$, $\mathbf{z}^*$ still retains sufficient min-entropy. Denote $\mathbf{z}_i$ as the intermediate evaluation on input $\mathbf{x}_i$ that is fed into $\mathsf{H}_{\mathsf{wise}}$. Then, we have the following:

$$\begin{aligned}
H_\infty(\mathbf{z}^* \mid \bar{\mathbf{A}}, \mathbf{F}, \mathbf{E}, \mathbf{R}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, S, (\mathbf{z}_i)_{i \in [Q]}) &= H_\infty(\mathbf{z}^* \mid \bar{\mathbf{A}}, \mathbf{F}, \mathbf{R}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, S, \mathbf{d}^\top \bar{\mathbf{A}}) & (18) \\
&= H_\infty(\mathbf{z}^* \mid \bar{\mathbf{A}}, \mathbf{F}, \mathbf{R}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, \mathbf{x}^*, \mathbf{d}^\top \bar{\mathbf{A}}) & (19) \\
&= H_\infty(\mathbf{z}^* - \mathbf{v}^* \mid \bar{\mathbf{A}}, \mathbf{F}, \mathbf{R}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, \mathbf{x}^*, \mathbf{d}^\top \bar{\mathbf{A}}) & \\
&\geq H_\infty(\mathbf{d} \mid \bar{\mathbf{A}}, \mathbf{F}, \mathbf{R}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, \mathbf{x}^*, \mathbf{d}^\top \bar{\mathbf{A}}) & \\
&= H_\infty(\mathbf{d} \mid \bar{\mathbf{A}}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, \mathbf{d}^\top \bar{\mathbf{A}}) & \\
&\geq H_\infty(\mathbf{d} \mid \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*) - h \cdot \log q. & (20)
\end{aligned}$$

Here, the first equality follows from $\mathbf{z}_i$ being uniquely determined by the values $\mathbf{x}_i, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, \mathbf{d}^\top \bar{\mathbf{A}}, \mathbf{F}, \mathbf{R}$, the second equality follows since $(\mathbf{x}_i)_{i \in [Q]}$ are now independent of $\mathbf{z}^*$, the forth inequality follows from $(q/p) \cdot (\mathbf{z}^* - \mathbf{v}^*)$ uniquely determining $\mathbf{d}$, and the last inequality follows since $\left|\mathbf{d}^\top \bar{\mathbf{A}}\right| = h \cdot \log q$. Finally, conditioned on event BAD_2 not occurring, $\mathbf{d}$ and $\mathbf{S}_{\mathbf{y}^*}$ are independent of each other. This is because fixing on $\mathbf{y}^*$, every entry of $\mathbf{S}_{\mathbf{y}^*}$ is distributed uniformly random over $[-(\bar{\beta} - \beta \cdot B), \bar{\beta} - \beta \cdot B]$, and in particular, independent of $\mathbf{d}$. Therefore, Equation (20) gives us

$$H_\infty(\mathbf{z}^* \mid \bar{\mathbf{A}}, \mathbf{F}, \mathbf{E}, \mathbf{R}, \mathbf{S}_{\mathbf{y}^*}, \mathbf{y}^*, S, (\mathbf{z}_i)_{i \in [Q]}) \geq n \cdot \log(2\beta + 1) - h \cdot \log q.$$

Finally, we finish the proof by arguing that we can extract statistically uniform bits from $\mathbf{z}^*$ by hashing it with a ζ-wise independent hash function $\mathsf{H}_{\mathsf{wise}}$. First of all, note that with all but negligible probability, the distribution of $\mathbf{z}^*$ is independent of the choice of $(\mathbf{x}_1, \cdots, \mathbf{x}_Q)$ (see Equation (18) and 19). In particular, the source $\mathbf{z}^*$ is taken from a distribution determined by $\mathbf{x}^*$ and there are at most $M := B^\ell$ possible choices for $\mathbf{x}^*$. Therefore, since $\mathbf{z}^*$ is chosen from a collection M distributions over $\mathbb{Z}_p^m$ of min-entropy at least $n \cdot \log(2\beta + 1) - h \cdot \log q$ each, we can apply Lemma 3.11 with $\epsilon = 2^{-\kappa}$ and our choice of ζ to conclude that $\mathsf{H}_{\mathsf{wise}}(\mathbf{z}^*)$ is negligibly close to $U(\mathbb{Z}_p^k)$. Hence, Game_6 is indistinguishable from Game_7. □

Lemma 5.9. *We have* $|\Pr[\mathsf{E}_7] - 1/2| = 0$.

Proof. In this game, the challenge is distributed equivalently for both $\mathsf{coin} = 0$ and 1. Therefore, no adversary has winning advantage in this game. □

Combining all the lemmas from Lemma 5.3, 5.5, 5.6, 5.7, 5.8 together, we have that $|\Pr[\mathsf{E}_7] - 1/2|$ is non-negligible. However, this contradicts Lemma 5.9. Hence, ϵ must be negligible and this concludes the proof. □

Theorem 5.10. *The above CPRF* Π_{CPRF} *for the inner product predicate* $\mathcal{C}^{\mathsf{IP}}$ *satisfies statistical weak* 1*-key privacy.*

Proof. Notice that a constrained key for a vector $\mathbf{y} \in [-B, B]^\ell$ is of the form:

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

where $\mathbf{S} \xleftarrow{\$} [\bar{\beta}, \bar{\beta}]^{n \times \ell}$ and $\mathbf{d} \xleftarrow{\$} [-\beta, \beta]^n$. We define the event BAD_2 similarly to in the proof of Theorem 5.1. That is, BAD_2 is the event that $\|\mathbf{S}_\mathbf{y}\|_\infty > \bar{\beta} - \beta \cdot B$ holds. As seen in the proof of Lemma 5.7, we have $\Pr[\mathsf{BAD}_2] = \mathsf{negl}(\kappa)$. Moreover, as seen in the proof of Lemma 5.8, conditioned on BAD_2 not occuring, every entry of $\mathbf{S}_\mathbf{y}$ is distributed uniformly random over $[-(\bar{\beta} - \beta \cdot B), \bar{\beta} - \beta \cdot B]$. Therefore, for any fixed $\mathbf{y} \in [-B, B]^\ell$, the distribution of $\mathsf{K}_\mathbf{y}$ is statistically close to the uniform distribution over $[-(\bar{\beta} - \beta \cdot B), \bar{\beta} - \beta \cdot B]^{n \times \ell}$. In particular, for any two vectors $\mathbf{y}_0$ and $\mathbf{y}_1$, the distributions of corresponding constrainbed keys $\mathsf{K}_{\mathbf{y}_0}$ and $\mathsf{K}_{\mathbf{y}_1}$ are statistically close. □

6 CPRF for $\mathsf{P/poly}$

6.1 Shift-Hiding Shiftable Function

Here, we review the notion of shift-hiding shiftable function (SHSF) introduced by Peikert and Shiehian [PS18]. 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 6.2.

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 $\Pi_{\mathsf{SHSF}} = (\mathsf{SHSF.KeyGen}, \mathsf{SHSF.Eval}, \mathsf{SHSF.Shift}, \mathsf{SHSF.SEval}, \mathsf{SHSF.Sim})$[9] 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.

[9] In the original definition of [PS18], 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.

$\mathsf{SHSF.Eval}(\mathsf{msk}, x) \to \mathbf{y}$: 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

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

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 $\mathsf{sk} \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}, C)$.
- 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 6.2 ([PS18]). *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 [PS18] 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

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

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

$$(\mathsf{SHSF.SEval}(\mathsf{sk}_C, x) + \mathbf{v})_i \notin \frac{q}{p}\mathbb{Z} + [-B, B]. \tag{21}$$

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.[10] The modified SHSF.Eval and SHSF.SEval will now add $\mathbf{r}$ to the original outputs, e.g., run SHSF.Eval of [PS18] 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 (21) 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 (21) 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. □

6.2 Construction of CPRF

Here, we give a construction of an adaptively secure CPRF for all polynomial-size circuits (i.e., $\mathsf{P/poly}$) 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 6.3. *If there exists a CPRF for* $\mathsf{P/poly}$ *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" ([Jag15, Yam17]). 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 Appendix D.

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 6.3, 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.[11]

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.[12]

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:[13]

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

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

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

[13] In our scheme, a public parameter is just the security parameter. So we omit the setup algorithm CPRF.Setup.

```
$\mathsf{ConstrainedKey}[\mathsf{sk}_1, C]$
Input: $x \in \{0,1\}^\ell$
Constants:$\mathsf{sk}_1$, $C$
If $C(x) = 1$
  Output $\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_1, x) \rfloor_p$
Else
  Output $\bot$
```

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

```
$\mathsf{ConstrainedKey\text{-}Pre}_k[\{\mathsf{msk}_i\}_{i\in[k]}, \{C_i\}_{i\in[k]}]$
Input: $x \in \{0,1\}^\ell$
Constants:$\{\mathsf{msk}_i\}_{i\in[k]}, \{C_i\}_{i\in[k]}$
If $C_k(x) = 1$
  Output $\lfloor \sum_{i\in[k]} \left(\prod_{i'\in[i-1]} \overline{C}_{i'}(x)\right) \cdot \mathsf{SHSF.Eval}(\mathsf{msk}_i, x) \rfloor_p$
Else
  Output $\bot$
```

Figure 4: Description of Program $\mathsf{ConstrainedKey\text{-}Pre}_k[\{\mathsf{msk}_i\}_{i\in[k]}, \{C_i\}_{i\in[k]}]$

$\mathsf{CPRF.Gen}(1^\kappa)$: 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 6.4. *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 6.2 and 6.3 we obtain the following theorem.

Theorem 6.5. *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 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 $\mathsf{P/poly}$.*

6.3 Security Proof

In this subsection, we prove Theorem 6.4. Let $\mathcal{A}$ be an adversary against the adaptive security of the CPRF that makes at most Q_1 and Q_2 pre-challenge and post-challenge constrained key queries, respectively. In the following, C_i denotes $\mathcal{A}$'s i-th key query for $i \in [Q_1 + Q_2]$. We consider the following sequence of games between $\mathcal{A}$ and the challenger. Below, let E_i denote the event that $\mathcal{A}$ correctly guesses coin (i.e., $\widehat{\mathsf{coin}} = \mathsf{coin}$) in Game_i.

Game_0 : This is the original adaptive security game. Specifically, the challenger generates $\mathsf{sk}_1 \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^{\sigma_1})$. It answers $\mathcal{A}$'s queries as follows:[14]

[14]Recall that we assume that $\mathcal{A}$ does not make an evaluation query.

```
$\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i\in[j]}, \{C_i\}_{i\in[j]}, \mathsf{sk}_{j+1}, C]$
Input: $x \in \{0,1\}^\ell$
Constants: $\{\mathsf{msk}_i\}_{i\in[j]}, \{C_i\}_{i\in[j]}, \mathsf{sk}_{j+1}, C$
If $C(x) = 1$
  Output $\lfloor \left(\prod_{i\in[j]} \overline{C}_i(x)\right) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{j+1}, x) + \sum_{i\in[j]} \left(\prod_{i'\in[i-1]} \overline{C}_{i'}(x)\right) \cdot \mathsf{SHSF.Eval}(\mathsf{msk}_i, x)\rfloor_p$
Else
  Output $\perp$
```

Figure 5: Description of Program $\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i\in[j]}, \{C_i\}_{i\in[j]}, \mathsf{sk}_{j+1}, C]$

```
$\mathsf{ConstrainedKey\text{-}Post\text{-}Alt}[\{\mathsf{msk}_i\}_{i\in[Q_1+1]}, \{C_i\}_{i\in[Q_1]}, C]$
Input: $x \in \{0,1\}^\ell$
Constants: $\{\mathsf{msk}_i\}_{i\in[Q_1+1]}, \{C_i\}_{i\in[Q_1]}, C$
If $C(x) = 1$
  Output $\lfloor \sum_{i\in[Q_1+1]} \left(\prod_{i'\in[i-1]} \overline{C}_{i'}(x)\right) \cdot \mathsf{SHSF.Eval}(\mathsf{msk}_i, x)\rfloor_p$
Else
  Output $\perp$
```

Figure 6: Description of Program $\mathsf{ConstrainedKey\text{-}Post\text{-}Alt}[\{\mathsf{msk}_i\}_{i\in[Q_1]}, \{C_i\}_{i\in[Q_1]}, C]$

Pre-challenge key query: When $\mathcal{A}$ makes a key query C, the challenger returns $\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey}[\mathsf{sk}_1, C])$.

Challenge query: When $\mathcal{A}$ makes a challenge query x^*, the challenger samples $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, and returns $\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_1, x^*)\rfloor_p$ if $\mathsf{coin} = 0$, and returns a uniformly random vector in $\mathbb{Z}_p^m$ otherwise.

Post-challenge key query: When $\mathcal{A}$ makes a key query C, the challenger returns $\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey}[\mathsf{sk}_1, C])$.

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

Game_1 : The challenger generates $\mathsf{msk}_i \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^{\sigma_i})$ for $i \in [Q_1]$. It answers $\mathcal{A}$'s queries as follows:

Pre-challenge key query: When $\mathcal{A}$ makes its k-th pre-challenge key query C_k, the challenger returns

$$\mathsf{K}_{C_k} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Pre}_k[\{\mathsf{msk}_i\}_{i\in[k]}, \{C_i\}_{i\in[k]}])$$

where $\mathsf{ConstrainedKey\text{-}Pre}_k[\{\mathsf{msk}_i\}_{i\in[k]}, \{C_i\}_{i\in[k]}]$ is a program described in Figure 4.

Challenge query: When $\mathcal{A}$ makes a challenge query x^*, the challenger generates $\mathsf{sk}_{Q_1+1} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^{\sigma_{Q_1+1}})$ and samples $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. Then it returns

$$\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{Q_1+1}, x^*) + \sum_{i\in[Q_1]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*)\rfloor_p$$

if $\mathsf{coin} = 0$, and returns a uniformly random vector in $\mathbb{Z}_p^m$ otherwise.

Post-challenge key query: When $\mathcal{A}$ makes a post-challenge key query C, the challenger returns

$$\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_{Q_1}[\{\mathsf{msk}_i\}_{i\in[Q_1]}, \{C_i\}_{i\in[Q_1]}, \mathsf{sk}_{Q_1+1}, C]).$$

where $\mathsf{ConstrainedKey\text{-}Post}_{Q_1}[\{\mathsf{msk}_i\}_{i\in[Q_1]}, \{C_i\}_{i\in[Q_1]}, \mathsf{sk}_{Q_1+1}, C]$ is described in Figure 5.[15]

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

[15] The program is defined for general j in Figure 5, and we just set $j := Q_1$ here. The program with other values of j will appear in the proof of Lemma 6.6 where we prove the indistinguishability between Game_0 and Game_1.

$\mathsf{Game}_2:$ This game is identical to the previous game except that sk_{Q_1+1} is generated as $\underline{\mathsf{sk}_{Q_1+1} \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}_{Q_1+1}, C_{\mathsf{eq}}[x^*, \mathbf{r}])}$ where $\mathsf{msk}_{Q_1+1} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^{\sigma_{Q_1+1}})$ and $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^m$. (Recall that $C_{\mathsf{eq}}[x^*, \mathbf{r}]$ denotes a circuit that outputs $\mathbf{r}$ on input $x = x^*$, and 0 otherwise.)

$\mathsf{Game}_3:$ This game is identical to the previous game except that the challenge query is responded by

$$\underline{\lfloor \mathbf{r} + \sum_{i \in [Q_1+1]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p}$$

in the case of $\mathsf{coin} = 0$ and post-challenge key queries are responded by

$$\underline{\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post\text{-}Alt}[\{\mathsf{msk}_i\}_{i \in [Q_1+1]}, \{C_i\}_{i \in [Q_1]}, C]).}$$

$\mathsf{Game}_4:$ This game is identical to the previous game except that the challenge query is responded by a uniformly random vector in $\mathbb{Z}_p^m$ regardless of the value of coin.

This completes the description of games. Clearly, we have $|\Pr[\mathsf{E}_4] - 1/2| = 0$. Therefore what we have to prove is that $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_4]| = \mathsf{negl}(\kappa)$. We prove this by the following lemmata.

Lemma 6.6. *We have* $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$ *assuming that* iO *is a secure indistinguishability obfuscator and* Π_{SHSF} *satisfies* p*-rounded* ϵ*-correctness and the shift hiding,*

Proof. For proving this lemma, we introduce further hybrids $\mathsf{Game}_{0.j.0}$ for $j \in \{0, ..., Q_1\}$ and $\mathsf{Game}_{0.j.1}$ for $j \in \{0, ..., Q_1 - 1\}$.

$\mathsf{Game}_{0.j.0}:$ In this game, the challenger first generates $\mathsf{msk}_i \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^{\sigma_i})$ for $i \in [j]$. Then it answers $\mathcal{A}$'s queries as follows:

Pre-challenge key query: When $\mathcal{A}$ makes its k-th pre-challenge key query C_k, the challenger returns K_{C_k} to $\mathcal{A}$ that is generated as follows.

- If $k \le j$, the challenger computes

$$\mathsf{K}_{C_k} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Pre}_k[\{\mathsf{msk}_i\}_{i \in [k]}, \{C_i\}_{i \in [k]}])$$

where $\mathsf{ConstrainedKey\text{-}Pre}_k[\{\mathsf{msk}_i\}_{i \in [k]}, \{C_i\}_{i \in [k]}]$ is a program described in Figure 4.

- Just after answering $\mathcal{A}$'s j-th key query, the challenger generates

$$\mathsf{sk}_{j+1} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^{\sigma_{j+1}}).$$

- If $k \ge j + 1$, the challenger computes

$$\mathsf{K}_{C_k} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i \in [j]}, \{C_i\}_{i \in [j]}, \mathsf{sk}_{j+1}, C_k])$$

where $\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i \in [j]}, \{C_i\}_{i \in [j]}, \mathsf{sk}_{j+1}, C_k]$ is a program described in Figure 5.

Challenge query: When $\mathcal{A}$ makes the challenge query x^*, the challenger samples $\mathsf{coin} \xleftarrow{\$} \{0, 1\}$ and returns

$$\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{j+1}, x^*) + \sum_{i \in [j]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p$$

if $\mathsf{coin} = 0$, and returns a uniformly random vector in $\mathbb{Z}_p^m$ otherwise.

Post-challenge key query: When $\mathcal{A}$ makes its key query C, the challenger computes

$$\mathsf{K}_{C_k} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i \in [j]}, \{C_i\}_{i \in [j]}, \mathsf{sk}_{j+1}, C])$$

where $\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i \in [j]}, \{C_i\}_{i \in [j]}, \mathsf{sk}_{j+1}, C]$ is a program described in Figure 5.

$\mathsf{Game}_{0.j.1}$: This game is identical to the previous game except that the generation of sk_{j+1} is delayed until $\mathcal{A}$ makes its $(j+1)$-th key query C_{j+1} and it is generated as

$$\underline{\mathsf{sk}_{j+1} \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}_{j+1}, \overline{C}_{j+1}(\cdot) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{j+2}, \cdot))}$$

where $\mathsf{msk}_{j+1} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^{\kappa}, 1^{\sigma_{j+1}})$ and $\mathsf{sk}_{j+2} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^{\kappa}, 1^{\sigma_{j+2}})$.

Note that $\mathsf{ConstrainedKey}[\mathsf{sk}_1, C]$ and $\mathsf{ConstrainedKey\text{-}Post}_0[\perp, \perp, \mathsf{sk}_1, C]$ are exactly the identical programs, Game_0 and $\mathsf{Game}_{0.0.0}$ are identical from $\mathcal{A}$'s view. It is also easy to see that Game_1 and $\mathsf{Game}_{0.Q_1,0}$ are identical from $\mathcal{A}$'s view.[16] Therefore what is left is to prove the following claims.

Claim 6.7. For $j \in \{0, ..., Q_1 - 1\}$, we have $|\Pr[\mathsf{E}_{0.j.0}] - \Pr[\mathsf{E}_{0.j.1}]| = \mathsf{negl}(\kappa)$ assuming that Π_{SHSF} satisfies the shift hiding,

Proof. The only difference between $\mathsf{Game}_{0.j.0}$ and $\mathsf{Game}_{0.j.1}$ is that sk_{j+1} is generated as $\mathsf{sk}_{j+1} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^{\kappa}, 1^{\sigma_{j+1}})$ in the former whereas it is generated as $\mathsf{sk}_{j+1} \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}_{j+1}, \overline{C}_{j+1}(\cdot) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{j+2}, \cdot))$ where $\mathsf{msk}_{j+1} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^{\kappa}, 1^{\sigma_{j+1}})$ and $\mathsf{sk}_{j+2} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^{\kappa}, 1^{\sigma_{j+2}})$ in the latter. The indistinguishability of these two distributions of sk_{j+1} is exactly what is required by the shift hiding property. Therefore it is straightforward to reduce the indistinguishability between these games to the shift hiding property of Π_{SHSF}. □

Claim 6.8. For $j \in \{0, ..., Q_1 - 1\}$, we have $|\Pr[\mathsf{E}_{0.j.1}] - \Pr[\mathsf{E}_{0.j+1.0}]| = \mathsf{negl}(\kappa)$ assuming that iO is a secure indistinguishability obfuscator and Π_{SHSF} satisfies p-rounded ϵ-correctness.

Proof. The differences between $\mathsf{Game}_{0.j.1}$ and $\mathsf{Game}_{0.j+1.0}$ are summarized as follows:

1. The challenge query is responded by $\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{j+1}, x^*) + \sum_{i \in [j]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p$ in the former whereas it is responded by $\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{j+2}, x^*) + \sum_{i \in [j+1]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p$ in the latter in the case of $\mathsf{coin} = 0$.

2. The $(j+1)$-th key query C_{j+1} is responded by $\mathsf{K}_{C_{j+1}} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i \in [j]}, \{C_i\}_{i \in [j]}, \mathsf{sk}_{j+1}, C_{j+1}])$ in the former whereas it is responded by $\mathsf{K}_{C_{j+1}} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Pre}_{j+1}[\{\mathsf{msk}_i\}_{i \in [j+1]}, \{C_i\}_{i \in [j+1]}])$ in the latter.

3. The k-th key query C_k for $k \geq j+2$ (which is either pre-challenge or post-challenge key query) is responded by $\mathsf{K}_{C_k} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i \in [j]}, \{C_i\}_{i \in [j]}, \mathsf{sk}_{j+1}, C_k])$ in the former whereas it is responded by $\mathsf{K}_{C_k} \xleftarrow{\$} \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_{j+1}[\{\mathsf{msk}_i\}_{i \in [j+1]}, \{C_i\}_{i \in [j+1]}, \mathsf{sk}_{j+2}, C_k])$ in the latter.

For proving these differences are computationally indistinguishable, we first show that we have

$$\begin{aligned} &\left\lfloor \left(\prod_{i \in [j]} \overline{C}_i(x) \right) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{j+1}, x) + \sum_{i \in [j]} \left(\prod_{i' \in [i-1]} \overline{C}_{i'}(x) \right) \cdot \mathsf{SHSF.Eval}(\mathsf{msk}_i, x) \right\rceil_p \\ &= \left\lfloor \left(\prod_{i \in [j+1]} \overline{C}_i(x) \right) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{j+2}, x) + \sum_{i \in [j+1]} \left(\prod_{i' \in [i-1]} \overline{C}_{i'}(x) \right) \cdot \mathsf{SHSF.Eval}(\mathsf{msk}_i, x) \right\rceil_p \end{aligned} \tag{22}$$

for all $x \in \{0,1\}^{\ell}$ with overwhelming probability. Assuming Equation (22) is correct for all $x \in \{0,1\}^{\ell}$, we can prove that the above three differences are computationally indistinguishable from $\mathcal{A}$'s view:

1. Since we must have $C_i(x^*) = 0$ for all i, Equation (22) implies $\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{j+1}, x^*) + \sum_{i \in [j]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p = \lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{j+2}, x^*) + \sum_{i \in [j+1]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p$. Therefore, the response to the challenge query does not differ between these games.

[16] Note that we stop the game hop at $\mathsf{Game}_{0.Q_1.0}$ instead of $\mathsf{Game}_{0.Q_1.1}$, which is not defined.

2. Noting that the program $\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i\in[j]}, \{C_i\}_{i\in[j]}, \mathsf{sk}_{j+1}, C_{j+1}]$ returns $\perp$ on any input x such that $C_{j+1}(x) = 0$, Equation (22) implies outputs of $\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i\in[j]}, \{C_i\}_{i\in[j]}, \mathsf{sk}_{j+1}, C_{j+1}]$ and $\mathsf{ConstrainedKey\text{-}Pre}_{j+1}[\{\mathsf{msk}_i\}_{i\in[j+1]}, \{C_i\}_{i\in[j+1]}]$ are the same for all inputs $x \in \{0,1\}^\ell$. Therefore, obfuscation of these two programs are computationally indistinguishable by security of iO.

3. Equation (22) immediately implies that outputs of programs $\mathsf{ConstrainedKey\text{-}Post}_j[\{\mathsf{msk}_i\}_{i\in[j]}, \{C_i\}_{i\in[j]}, \mathsf{sk}_{j+1}, C_k]$ and $\mathsf{ConstrainedKey\text{-}Post}_{j+1}[\{\mathsf{msk}_i\}_{i\in[j+1]}, \{C_i\}_{i\in[j+1]}, \mathsf{sk}_{j+2}, C_k]$ are the same for all inputs $x \in \{0,1\}^\ell$. Therefore, obfuscation of these two programs are computationally indistinguishable by security of iO.

Then we can conclude that these two games are computationally indistinguishable.

What is left is to prove that Equation (22) holds for all $x \in \{0,1\}^\ell$ with overwhelming probability. Here, we fix $x \in \{0,1\}^\ell$, $\{C_i\}_{i\in[j+1]}$, $\{\mathsf{msk}_i\}_{i\in[j]}$, and sk_{j+2}. If $\prod_{i\in[j]} \overline{C}_i(x) = 0$, it is immediate to see that Equation (22) holds. If $\prod_{i\in[j]} \overline{C}_i(x) = 1$, then the left hand side can be written as

$$\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{j+1}, x) + \mathbf{v} \rceil_p$$

where we set $\mathbf{v} := \sum_{i\in[j]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x)$. Applying the p-rounded ϵ-correctness, this is equal to

$$\lfloor \mathsf{SHSF.Eval}(\mathsf{msk}_{j+1}, x) + \overline{C}_{j+1}(x) \cdot \mathsf{SHSF.SEval}(\mathsf{sk}_{j+2}, x) + \mathbf{v} \rceil_p$$

except probability ϵ where the probability is taken over the choice of msk_{j+1} and sk_{j+1}. Noting that we assumed $\prod_{i\in[j]} \overline{C}_i(x) = 1$, this is equal to the right hand side of Equation (22). So far, we have shown that Equation (22) holds with probability except ϵ for any fixed $x \in \{0,1\}^\ell$, $\{C_i\}_{i\in[j+1]}$, $\{\mathsf{msk}_i\}_{i\in[j]}$, and sk_{j+2}. Noting that distributions of $\{C_i\}_{i\in[j+1]}$, $\{\mathsf{msk}_i\}_{i\in[j]}$, and sk_{j+2} are independent from those of msk_{j+1} and sk_{j+1},[17] by taking the union bound for all $x \in \{0,1\}^\ell$, the probability that there exists $x \in \{0,1\}^\ell$ such that Equation (22) does not hold is at most $2^\ell\epsilon = \mathsf{negl}(\kappa)$ since we set $\epsilon := 2^{-\ell}\mathsf{negl}(\kappa)$. This completes the proof of Claim 6.8. □

Combining Claims 6.7 and 6.8, the proof of Lemma 6.6 is completed. □

Lemma 6.9. *We have* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$ *assuming that* Π_{SHSF} *satisfies the shift hiding.*

Proof. The only difference between Game_1 and Game_2 is that sk_{Q_1+1} is generated as $\mathsf{sk}_{Q_1+1} \xleftarrow{\$} \mathsf{SHSF.Sim}(1^\kappa, 1^{\sigma_{Q_1+1}})$ in Game_1 whereas it is generated as $\mathsf{sk}_{Q_1+1} \xleftarrow{\$} \mathsf{SHSF.Shift}(\mathsf{msk}_{Q_1+1}, C_{\mathsf{eq}}[x^*, \mathbf{r}])$ where $\mathsf{msk}_{Q_1+1} \xleftarrow{\$} \mathsf{SHSF.KeyGen}(1^\kappa, 1^{\sigma_{Q_1+1}})$ and $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^m$ in Game_2. The indistinguishability of these two distributions of sk_{Q_1+1} is what is exactly required by the shift hiding property of SHSF. Thus it is straightforward to reduce the indistinguishability of these two games to the shift hiding property of Π_{SHSF}. □

Lemma 6.10. *We have* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\kappa)$ *assuming that* iO *is a secure indistinguishability obfuscator and* Π_{SHSF} *satisfies p-rounded ϵ-correctness.*

Proof. The proof of this lemma is similar to that of Claim 6.8. The differences between Game_2 and Game_3 are summarized as follows:

1. The challenge query is responded by $\lfloor \mathsf{SHSF.SEval}(\mathsf{sk}_{Q_1+1}, x^*) + \sum_{i\in[Q_1]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p$ in the former whereas it is responded by $\lfloor \mathbf{r} + \sum_{i\in[Q_1+1]} \mathsf{SHSF.Eval}(\mathsf{msk}_i, x^*) \rceil_p$ in the latter in the case of $\mathsf{coin} = 0$.

2. Post-challenge key queries are responded by $\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_{Q_1}[\{\mathsf{msk}_i\}_{i\in[Q_1]}, \{C_i\}_{i\in[Q_1]}, \mathsf{sk}_{Q_1+1}, C])$ in the former whereas it is responded by $\mathsf{K}_C := \mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post\text{-}Alt}[\{\mathsf{msk}_i\}_{i\in[Q_1+1]}, \{C_i\}_{i\in[Q_1]}, C])$ in the latter.

[17] Note that no information of msk_{j+1} and sk_{j+1} is given to $\mathcal{A}$ before it makes $(j+1)$-th query C_{j+1}.

For proving that these differences are computationally indistinguishable from $\mathcal{A}$'s view, we show that we have

$$\begin{aligned}&\lfloor\left(\prod_{i\in[Q_1]}\overline{C}_i(x)\right)\cdot\mathsf{SHSF.SEval}(\mathsf{sk}_{Q_1+1},x)+\sum_{i\in[Q_1]}\left(\prod_{i'\in[i-1]}\overline{C}_{i'}(x)\right)\cdot\mathsf{SHSF.Eval}(\mathsf{msk}_i,x)\rfloor_p\\ &\quad=\lfloor\left(\prod_{i\in[Q_1]}\overline{C}_i(x)\right)\cdot C_{\mathsf{eq}}[x^*,\mathbf{r}](x)+\sum_{i\in[Q_1+1]}\left(\prod_{i'\in[i-1]}\overline{C}_{i'}(x)\right)\cdot\mathsf{SHSF.Eval}(\mathsf{msk}_i,x)\rfloor_p \qquad (23)\end{aligned}$$

Assuming Equation (23) is correct for all $x\in\{0,1\}^\ell$, we can prove that the above two differences are computationally indistinguishable from $\mathcal{A}$'s view:

1. Since we must have $C_i(x^*)=0$ for all i and we have $C_{\mathsf{eq}}[x^*,\mathbf{r}](x^*)=\mathbf{r}$ by definition, Equation (23) implies $\lfloor\mathsf{SHSF.SEval}(\mathsf{sk}_{Q_1+1},x^*)+\sum_{i\in[Q_1]}\mathsf{SHSF.Eval}(\mathsf{msk}_i,x^*)\rfloor_p=\lfloor\mathbf{r}+\sum_{i\in[Q_1+1]}\mathsf{SHSF.Eval}(\mathsf{msk}_i,x^*)\rfloor_p$. Therefore, the response to the challenge query does not differ between these games.

2. For any post-challenge key query C_k, we must have $C_k(x^*)=0$ by the definition of the security game of CPRF. Therefore for any $x\in\{0,1\}^\ell$, if $\mathsf{K}_C:=\mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_{Q_1}[\{\mathsf{msk}_i\}_{i\in[Q_1]},\{C_i\}_{i\in[Q_1]},\mathsf{sk}_{Q_1+1},C])$ does not output $\perp$ on input x, then we have $x\neq x^*$ and thus $C_{\mathsf{eq}}[x^*,\mathbf{r}](x)=0$. Based on this observation, Equation (23) implies that outputs of programs $\mathsf{K}_C:=\mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post}_{Q_1}[\{\mathsf{msk}_i\}_{i\in[Q_1]},\{C_i\}_{i\in[Q_1]},\mathsf{sk}_{Q_1+1},C])$ and $\mathsf{K}_C:=\mathsf{iO}(\mathsf{ConstrainedKey\text{-}Post\text{-}Alt}[\{\mathsf{msk}_i\}_{i\in[Q_1+1]},\{C_i\}_{i\in[Q_1]},C])$ are the same for all inputs $x\in\{0,1\}^\ell$. Therefore, obfuscation of these two programs are computationally indistinguishable by security of iO.

Then we can conclude that these two games are computationally indistinguishable.

What is left is to prove that Equation (23) holds for all $x\in\{0,1\}^\ell$ with overwhelming probability. Here, we fix $x\in\{0,1\}^\ell$, $\{C_i\}_{i\in[Q_1]}$, $\{\mathsf{msk}_i\}_{i\in[Q_1]}$ and $\mathbf{r}$. If $\prod_{i\in[Q_1]}\overline{C}_i(x)=0$, it is immediate to see that Equation (23) holds. If $\prod_{i\in[Q_1]}\overline{C}_i(x)=1$, then the left hand side can be written as

$$\lfloor\mathsf{SHSF.SEval}(\mathsf{sk}_{Q_1+1},x)+\mathbf{v}\rfloor_p$$

where we set $\mathbf{v}:=\sum_{i\in[Q_1]}\mathsf{SHSF.Eval}(\mathsf{msk}_i,x)$. Applying the p-rounded ϵ-correctness, this is equal to

$$\lfloor\mathsf{SHSF.Eval}(\mathsf{msk}_{Q_1+1},x)+C_{\mathsf{eq}}[x^*,\mathbf{r}](x)+\mathbf{v}\rfloor_p$$

except probability ϵ where the probability is taken over the choice of msk_{Q_1+1} and sk_{Q_1+1}. Noting that we assumed $\prod_{i\in[Q_1]}\overline{C}_i(x)=1$, this is equal to the right hand side of Equation (23). So far, we have shown that Equation (23) holds with probability except ϵ for any fixed $x\in\{0,1\}^\ell$, $\{C_i\}_{i\in[Q_1]}$, $\{\mathsf{msk}_i\}_{i\in[Q_1]}$ and $\mathbf{r}$. Noting that distributions of $\{C_i\}_{i\in[Q_1]}$, $\{\mathsf{msk}_i\}_{i\in[Q_1]}$ and $\mathbf{r}$ are independent from those of msk_{Q_1+1} and sk_{Q_1+1},[18] by taking the union bound for all possible $x\in\{0,1\}^\ell$, the probability that there exists $x\in\{0,1\}^\ell$ such that Equation (23) does not hold is at most $2^\ell\epsilon=\mathsf{negl}(\kappa)$ since we set $\epsilon:=2^{-\ell}\mathsf{negl}(\kappa)$. This completes the proof of Lemma 6.10. □

Lemma 6.11. *We have* $|\Pr[\mathsf{E}_3]-\Pr[\mathsf{E}_4]|=\mathsf{negl}(\kappa)$ *assuming* $p<\mathsf{negl}(\kappa)\cdot q$.

Proof. Due to the modification made in Game_3, sk_{Q_1+1} is not used at all in this game. Thus, information of $\mathbf{r}$ is leaked only through the challenge oracle. Therefore, we can replace the challenge value with $\lfloor\mathbf{r}'\rfloor_p$ for $\mathbf{r}'\overset{\$}{\leftarrow}\mathbb{Z}_q^m$. Since we have $p<\mathsf{negl}(\kappa)\cdot q$, the distribution of $\lfloor\mathbf{r}'\rfloor_p$ is statistically close to the uniform distribution over $\mathbb{Z}_p^m$. □

Combining Lemmata 6.6 and 6.9 to 6.11 and $|\Pr[\mathsf{E}_4]-1/2|=0$, we obtain $|\Pr[\mathsf{E}_0]-1/2|=\mathsf{negl}(\kappa)$, which concludes the proof of Theorem 6.4.

[18] Note that no information of msk_{Q_1+1} and sk_{Q_1+1} is given to $\mathcal{A}$ before it makes Q_1-th query C_{Q_1}.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010. (Cited on page 3.)

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2013. (Cited on page 9, 12.)

[AMN$^+$18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 543–574. Springer, Heidelberg, August 2018. (Cited on page 2, 3, 4, 42, 43.)

[AMN$^+$19] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively single-key secure constrained PRFs for NC^1. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 223–253. Springer, Heidelberg, April 2019. (Cited on page 3, 4, 42, 43.)

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, August 2004. (Cited on page 9.)

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 31–60. Springer, Heidelberg, March 2015. (Cited on page 2, 16, 17, 42, 43.)

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. (Cited on page 8, 9, 13.)

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012. (Cited on page 14.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 2, 42, 43.)

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 567–594. Springer, Heidelberg, November 2017. (Cited on page 42.)

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 415–445. Springer, Heidelberg, April / May 2017. (Cited on page 8.)

[BKPW12] Mihir Bellare, Eike Kiltz, Chris Peikert, and Brent Waters. Identity-based (lossy) trapdoor functions and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 228–245. Springer, Heidelberg, April 2012. (Cited on page 9.)

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013. (Cited on page 7, 8.)

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. (Cited on page 12.)

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 494–524. Springer, Heidelberg, March 2017. (Cited on page 3, 16, 17, 43.)

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Heidelberg, August 2014. (Cited on page 7, 8.)

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012. (Cited on page 2.)

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 407–424. Springer, Heidelberg, April 2009. (Cited on page 13.)

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Heidelberg, November 2017. (Cited on page 2, 3, 8, 42, 43.)

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015. (Cited on page 2, 3, 8, 42, 43.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 2, 3, 4, 17, 42, 43.)

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014. (Cited on page 10, 43.)

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 446–476. Springer, Heidelberg, April / May 2017. (Cited on page 2, 3, 16, 42, 43.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010. (Cited on page 3.)

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018. (Cited on page 2, 3, 42, 43.)

[DN18] Alex Davidson and Ryo Nishimaki. A bit-fixing PRF with O(1) collusion-resistance from LWE. Cryptology ePrint Archive, Report 2018/890, 2018. https://eprint.iacr.org/2018/890. (Cited on page 1.)

[Dod00] Yevgeniy Dodis. *Exposure-resilient cryptography*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000. (Cited on page 14.)

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014. (Cited on page 4, 42.)

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. (Cited on page 14.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. (Cited on page 2, 5, 15, 17, 42, 44.)

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 537–566. Springer, Heidelberg, November 2017. (Cited on page 42.)

[GKPV10] Shafi Goldwasser, Yael Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. *ICS*, pages 230–240, 2010. (Cited on page 9.)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (Cited on page 5, 15, 17, 44.)

[HKKW19] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 357–376. Springer, Heidelberg, February 2019. (Cited on page 2, 4, 42, 43.)

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 79–102. Springer, Heidelberg, November / December 2015. (Cited on page 2, 3, 4, 42, 43.)

[Jag15] Tibor Jager. Verifiable random functions from weaker assumptions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 121–143. Springer, Heidelberg, March 2015. (Cited on page 13, 31.)

[JKK$^+$17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, August 2017. (Cited on page 42.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013. (Cited on page 2, 42, 43.)

[KY16] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Heidelberg, December 2016. (Cited on page 13.)

[KY18] Shuichi Katsumata and Shota Yamada. Note on constructing constrained PRFs from OWFs with constant collusion resistance. *IACR Cryptology ePrint Archive*, 2018:914, 2018. (Cited on page 1.)

[KY19] Shuichi Katsumata and Shota Yamada. Non-zero inner product encryption schemes from various assumptions: LWE, DDH and DCR. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 158–188. Springer, Heidelberg, April 2019. (Cited on page 3.)

[LSSS17] Benoît Libert, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 332–364. Springer, Heidelberg, August 2017. (Cited on page 9.)

[LST18] Benoît Libert, Damien Stehlé, and Radu Titiu. Adaptively secure distributed PRFs from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 391–421. Springer, Heidelberg, November 2018. (Cited on page 9, 12, 13.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012. (Cited on page 8, 28.)

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. (Cited on page 2.)

[NRR02] Moni Naor, Omer Reingold, and Alon Rosen. Pseudorandom functions and factoring. *SIAM J. Comput.*, 31(5):1383–1404, 2002. (Cited on page 2.)

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009. (Cited on page 12.)

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 675–701. Springer, Heidelberg, March 2018. (Cited on page 2, 3, 4, 11, 29, 30, 31, 42, 43.)

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. (Cited on page 11, 12.)

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. (Cited on page 10.)

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Heidelberg, August 2019. (Cited on page 3, 4.)

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005. (Cited on page 7, 20.)

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 161–193. Springer, Heidelberg, August 2017. (Cited on page 13, 31.)

A Related Works

Predicates. Before explaining existing constructions of CPRFs, we briefly review some special classes of predicates to illustrate the types of predicates that are of interest. Let $x = x_1 \dots x_\ell \in \{0,1\}^\ell$ be a PRF input, and let $x|_{l_1}^{l_2} = x_{l_1} \dots x_{l_2}$. The following predicates are "loosely" listed in ascending order of expressibility:[19]

- puncturing: $C_v(x) = 1$ iff $x \neq v$;

[19] We say loosely since some classes are incomparable. For instance, the class of prefix-fixing is not necessarily more expressive than puncturing.

- prefix-fixing: $C_v(x) = 1$ iff $x|_l = v$ where $l \leq \ell$;
- left-right-fixing: $C_{v^b}(x) = 1$ iff $x|_{b \cdot l}^{(b+1)\cdot l} = v^b$ where $b \in \{0,1\}$, $\ell = 2l$, and $v^b \in \{0,1\}^l$;
- bit-fixing: $C_v(x) = 1$ iff $(x_i = v_i) \vee (v_i = *)$ for each $i \in [\ell]$, where $v_i \in \{0,1,*\}^\ell$;
- t-conjunctive normal form (t-CNF): $C^{t\text{-cnf}}(x) = 1$ iff $C^{t\text{-cnf}}(x) = \wedge_i C_i(x)$ and $C_i \in \mathsf{NC}_t^0$ where NC_t^0 is the class of NC^0 circuits that read at most t indices of the input;
- inner-product: $C_{\mathbf{v}}(\mathbf{x}) = 1$ iff $\langle \mathbf{v}, \mathbf{x} \rangle = 0$ where $\mathbf{x}, \mathbf{v}$ resides in a space where inner-product is defined;
- general circuits in $\{\mathsf{NC}^1, \mathsf{P/poly}\}$: $C(x) = 1$ where $C \in \{\mathsf{NC}^1, \mathsf{P/poly}\}$.

The expressibility of the inner-product predicate is between that of t-CNF and NC^1. In this study, the predicates that we will be considering are the bit-fixing, t-CNF, inner-product, and $\mathsf{P/poly}$ predicates. Note that bit-fixing predicates are strictly included in t-CNF predicates for $t = 1$. We explicitly include bit-fixing predicates for ease of presentation and understanding.

A.1 Existing Constructions

Since the original works of CPRFs [BW13, KPTZ13, BGI14], numerous constructions of CPRFs have been given, relying on different primitives and providing a range of functionality. We summarize, to the best of our knowledge, all known constructions in Table 2. In the table, abbreviations and remarks are as follows: 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, MLDDH, 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. We do not consider the CPRFs of [Bit17, GHKW17] since they do not permit evaluation queries. Regarding key privacy, † means that they did not originally consider key-privacy.

In the following, we provide a survey on known CPRFs.

Adaptively secure constructions. It was observed in the original works that the GGM-PRF [GGM86] can be used as a CPRF for puncturing or prefix-fixing predicates [BW13, KPTZ13, BGI14]. While these original works were unable to achieve adaptive security (without sub-exponential security losses), the work of Fuchsbauer et al. [FKPR14] show that the proof technique can be modified to achieve adaptive security with a quasi-polynomial security loss. This proof technique was adapted and simplified by Jafargholi et al. [JKK+17]. However, it is still an open problem whether the CPRF for prefix-fixing predicates is adaptively secure under a polynomial-reduction loss. Boneh and Waters present adaptively secure CPRF for left-right-fixing predicates based on a pairing-based assumption in the ROM [BW13]. Hohenberger et al. present adaptively secure and collusion-resistant CPRFs for $\mathsf{P/poly}$ based on IO in the ROM [HKKW19]. Hohenberger et al. present adaptively secure and collusion-resistant puncturable CPRFs based on IO and sub-group decision assumption [HKW15]. Attrapadung et al. present adaptively secure and single-key CPRFs for bit-fixing based on the ROM, and NC^1 based on the ROM and a non-static assumption [AMN+18]. In another work, the same authors present adaptively secure and single-key CPRFs for NC^1 based on IO and sub-group decision assumption [AMN+19]. Thus, even if we rely on IO, the state-of-the-art construction of adaptively secure CPRFs is single-key CPRF for NC^1 unless we rely on the ROM.

Selectively secure construction. At the early stage of research on CPRFs, most constructions were selectively secure [BW13, KPTZ13, BGI14, BFP+15] except for the CPRF for left-right predicates by Boneh and Waters [BW13]. There are several CPRFs for $\mathsf{P/poly}$ (or NC^1) based on the LWE assumption in the standard model. However, all of them are selectively secure [BV15, BTVW17, CC17, CVW18, PS18] and do not satisfy collusion-resistance. The work of Attrapadung et al. [AMN+18] provides CPRFs for bit-fixing and NC^1 from traditional groups. However, their constructions too do not satisfy adaptive security or collusion resistance without relying on the ROM.

Therefore, thus far, all known CPRF constructions from standard assumptions in the standard model do not achieve adaptive security (for expressive predicates) or collusion resistance (even for 2 keys).

Achieving Private Constraints. An additional security requirement that was introduced by Boneh et al. [BLW17] is that the constrained keys do not reveal the constraint that is encoded in them. In other words, given a constrained key for one of two adversarially-chosen constraints, the same adversary is unable to distinguish which constraint is encoded with more than a negligible advantage. A CPRF satisfying this definition of security is known as a private CPRF or PCPRF.[20] The CPRF for the prefix-fixing predicates based on the GGM-PRF [BW13, KPTZ13, BGI14] trivially achieves 1-key privacy. The constructions of [BLW17] satisfy poly-key privacy (hence poly constrained key queries) for circuit predicates under the existence of IO with sub-exponential security loss. The PCPRFs of [CC17, BTVW17, PS18, CVW18] satisfy 1-key privacy for circuit predicates. Achieving privacy for $m > 1$ seems challenging, since it would imply the existence of IO for $\mathsf{P/poly}$ from LWE [CC17]. Finally, CPRF for left-right fixing predicates shown in [BW13] satisfies poly-key privacy in the random oracle model and the CPRF for bit-fixing predicates shown in [AMN+18] satisfies 1-key privacy.

Table 2: List of existing constructions of CPRFs along with their functionality and the assumptions required.

	Adaptive	Collusion-resistance	Privacy	Predicate	Assumption
[BW13]	×	1	$1^{\dagger}$	Prefix	OWF
	✓	poly	poly	LR	BDDH & ROM
	×	poly	0	BF	MLDDH
	×	poly	0	$\mathsf{P/poly}$	MLDDH
[KPTZ13]	×	1	1	Prefix	OWF
[BGI14]	×	1	$1^{\dagger}$	Prefix	OWF
[BZ14]	×	poly	0	$\mathsf{P/poly}$	IO
[HKKW19]	✓	poly	0	$\mathsf{P/poly}$	IO & ROM
[BFP+15]	×	poly	0	Prefix	LWE
[BV15]	×	1	0	$\mathsf{P/poly}$	LWE
[HKW15]	✓	poly	0	Puncturing	SGH & IO
[BLW17]	×	poly	1 (weak)	Puncturing	MLDDH
	×	poly	1 (weak)	BF	MLDDH
	×	poly	poly	$\mathsf{P/poly}$	IO
[BTVW17]	×	1	1	$\mathsf{P/poly}$	LWE
[CC17]	×	1	1	BF	LWE
	×	1	1	NC^1	LWE
[AMN+18]	×	1	1	BF	DDH
	×	1	0	NC^1	L-DDHI
	✓	1	1	BF	ROM
	✓	1	0	NC^1	L-DDHI & ROM
[CVW18]	×	1	1	NC^1	LWE
[PS18]	×	1	1	$\mathsf{P/poly}$	LWE
[AMN+19]	✓	1	0	NC^1	SGH & IO
Section 4	✓	$O(1)$	$1(weak)$	t-CNF ($\supseteq$ BF)	OWF
Section 5	✓	1	$1(weak)$	IP	LWE
Section 6	✓	$O(1)$	0	$\mathsf{P/poly}$	LWE & IO

B CPRFs for t-CNF from Standard PRFs

In this section, we provide a construction of an adaptively pseudorandom on constrained points, Q-collusion resistant CPRFs for t-CNF predicates from any PRF, where Q can be set to be any constant independent of the security

[20] They are also known as 'constraint-hiding' CPRFs.

parameter. Similarly to the result in Section 4, our result implies the existence of such CPRFs from one-way functions [GGM86, HILL99].

B.1 Considered Predicate Class: t-CNF Predicates

The class of predicates which we will be using: t-conjunctive normal form (t-CNF). Informally, it contains the class of conjunction of $\mathbf{NC}^0$ circuits.

Definition B.1 (**t-CNF Predicates**)**.** *Let $\mathcal{S}$ denote the set $\mathcal{S} := \{(a_1, \cdots, a_t) \in [n]^t \mid a_1 < \cdots < a_t\}$. Then, a t-CNF predicate $C^{t\text{-cnf}} : \{0,1\}^n \to \{0,1\}$ such that $t \leq n$ is a set of clauses $C^{t\text{-cnf}} := \{(J, C_J)\}_{J \in \mathcal{S}}$ where $C_J : \{0,1\}^t \to \{0,1\}$. For all $x \in \{0,1\}^n$, a t-CNF predicate C is computed as follows:*

$$C^{t\text{-cnf}}(x) = \bigwedge_{J \in \mathcal{S}} C_J(x_J),$$

where $x_J \in \{0,1\}^t$ denotes the bit string consisting of the bits of x in the indices of J. Finally, a family of t-CNF predicate $\mathcal{C}_n^{t\text{-cnf}}$ is the set of t-CNF predicates with input length n.

Here, it is easy to see that the family of bit-fixing predicates (for length n inputs) $\mathcal{C}_n^{\mathsf{BF}}$ is included in $\mathcal{C}_n^{t\text{-cnf}}$ for any $t \geq 1$. In particular, when $t = 1$, we have $\mathcal{S} = [n]$. Therefore, each circuit $C_J : \{0,1\}^t \to \{0,1\}$ in the clause $\{(J, C_J)\}_{J \in [n]}$ only looks at the J^{th}-bit of the input $x \in \{0,1\}^n$ as required by the bit-fixing predicates. We note that we could have defined $\mathcal{C}^{t\text{-cnf}}$ to be a set of circuits of the form $\bigwedge_{J \in T} C_J$ for some $T \subseteq \mathcal{S}$, however, the above definition is without loss of generality because we can always add dummy circuits that output the constant 1 function so that C includes a circuit C_J for each $J \in \mathcal{S}$.

Similarly to the case in Section 4.1, we prepare a helper function $G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}$ which would come in handy during the construction; it is responsible for outputting the set of authorized inputs corresponding to a t-CNF predicate. Formally, the function $G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}$ takes as input a t-CNF predicate $C^{t\text{-cnf}} = \{(J, C_J)\}_{J \in \mathcal{S}}$ and a tuple $\mathbb{J} = (J_1, \cdots, J_Q) \in \mathcal{S}^Q$ and outputs a tuple $\mathbb{W} = (w_1, \cdots, w_Q) \in (\{0,1\}^t)^Q$. The value of $G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}$ is computed as

$$G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}(C^{t\text{-cnf}}, \mathbb{J}) = \Big\{ \mathbb{W} \in (\{0,1\}^t)^Q \mid \bigwedge_{i \in [Q]} C_{J_i}(w_i) = 1 \Big\}.$$

Here, with an abuse of notation, we define the function $G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}$ for all t-CNF predicate family and positive integer Q.

Remark B.2. In case $t = O(1)$ and $Q = O(1)$, the size of $\mathcal{S}^Q$ and $(\{0,1\}^t)^Q$ are both polynomial in n.

B.2 Construction

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

Our Q-collusion resistance CPRF Π_{CPRF} for the constrained class $\mathcal{C}^{t\text{-cnf}}$ is provided as follows:

$\mathsf{CPRF.Gen}(1^\kappa) \to \mathsf{K}$: On input the security parameter 1^κ, it runs $\mathsf{K}_{\mathbb{J},\mathbb{W}} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ and $\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ for all $\mathbb{J} \in \mathcal{S}^Q$ and $\mathbb{W} \in (\{0,1\}^t)^Q$. Then it outputs the master key as

$$\mathsf{K} = \Big((\mathsf{K}_{\mathbb{J},\mathbb{W}}), (\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}}) \Big)_{\mathbb{J} \in \mathcal{S}^Q, \mathbb{W} \in (\{0,1\}^t)^Q}.$$

Here, recall $\mathcal{S} := \{(a_1, \cdots, a_t) \in [n]^t \mid a_1 < \cdots < a_t\}$.

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

$$\left((\mathsf{K}_{\mathbb{J},\mathbb{W}}), (\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}})\right)_{\mathbb{J}\in\mathcal{S}^Q, \mathbb{W}\in(\{0,1\}^t)^Q} \leftarrow \mathsf{K}.$$

It then computes

$$y = \bigoplus_{\mathbb{J}\in\mathcal{S}^Q} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J},x_{\mathbb{J}}}, x),$$

where $x_{\mathbb{J}}$ is defined as the string $(x_{J_1}, \cdots x_{J_Q}) \in (\{0,1\}^t)^Q$, $\mathbb{J} = (J_1, \cdots, J_Q)$. Finally, it outputs $y \in \{0,1\}^k$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C^{t\text{-}\mathsf{cnf}}) :\to \mathsf{K}_C$: On input the master key K and a circuit $C^{t\text{-}\mathsf{cnf}} \in \mathcal{C}_n^{t\text{-}\mathsf{cnf}}$, it first parses $\left((\mathsf{K}_{\mathbb{J},\mathbb{W}}), (\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}})\right)_{\mathbb{J}\in\mathcal{S}^Q, \mathbb{W}\in(\{0,1\}^t)}$ K and $\{(J, C_J)\}_{J\in\mathcal{S}} \leftarrow C^{t\text{-}\mathsf{cnf}}$. Then it outputs the constrained key

$$\mathsf{K}_C = \left(\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}\right)_{\mathbb{J}\in\mathcal{S}^Q, \mathbb{W}\in(\{0,1\}^t)^Q},$$

where $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}} = \mathsf{K}_{\mathbb{J},\mathbb{W}}$ if $\mathbb{W} \in G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}}, \mathbb{J})$, and $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}} = \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}}$ otherwise. Here, recall that $G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}}, \mathbb{J}) = \{\mathbb{W} \in (\{0,1\}^t)^Q \mid \bigwedge_{i\in[Q]} C_{J_i}(w_i) = 1\}$.

$\mathsf{CPRF.ConstrainEval}(\mathsf{K}_C, x) :\to y$: On input the constrained key K_C and an input $x \in \{0,1\}^n$, it first parses $\left(\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}\right)_{\mathbb{J}\in\mathcal{S}^Q, \mathbb{W}\in(\{0,1\}^t)^Q} \leftarrow \mathsf{K}_C$. It then uses the PRF keys included in the constrained key and computes

$$y = \bigoplus_{\mathbb{J}\in\mathcal{S}^Q} \mathsf{PRF.Eval}(\widetilde{\mathsf{K}}_{\mathbb{J},x_{\mathbb{J}}}, x),$$

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

B.3 Correctness

We check correctness of our CPRF. Let $C^{t\text{-}\mathsf{cnf}}$ be any t-CNF predicate in $\mathcal{C}_n^{t\text{-}\mathsf{cnf}}$. Put differently, let us fix an arbitrary $\{(J, C_J)\}_{J\in\mathcal{S}}$. By construction we have

$$\mathsf{K}_C = \left(\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}\right)_{\mathbb{J}\in\mathcal{S}^Q, \mathbb{W}\in(\{0,1\}^t)^Q} \xleftarrow{\$} \mathsf{CPRF.Constrain}(\mathsf{K}, C^{t\text{-}\mathsf{cnf}}).$$

Now, for any $x \in \{0,1\}^n$ such that $C^{t\text{-}\mathsf{cnf}}(x) = 1$, by definition of the t-CNF predicate, we have

$$\bigwedge_{J\in\mathcal{S}} C_J(x_J) = 1.$$

In particular, for all $\mathbb{J} \in \mathcal{S}^Q$, we have $\bigwedge_{i\in[Q]} C_{J_i}(x_{J_i}) = 1$. Then, by definition of function $G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}$, we have $x_{\mathbb{J}} \in G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}}, \mathbb{J})$ for any $\mathbb{J} \in \mathcal{S}^Q$. Hence, we have

$$\widetilde{\mathsf{K}}_{\mathbb{J},x_{\mathbb{J}}} = \mathsf{K}_{\mathbb{J},x_{\mathbb{J}}} \in \mathsf{K}_C \text{ for all } \mathbb{J} \in \mathcal{S}^Q.$$

Therefore, since $\mathsf{CPRF.Eval}$ and $\mathsf{CPRF.ConstrainEval}$ are computed exactly in the same way, using the same PRF keys, correctness holds.

B.4 Pseudorandomness on Constrained Points

In this section we show security of our CPRF Π_{CPRF} for the t-CNF predicate $\mathcal{C}^{t\text{-}\mathsf{cnf}}$. The following proofs follow essentially the same structure as the proof in Theorem 4.2.

Theorem B.3. *The above CPRF Π_{CPRF} for the t-CNF predicate $\mathcal{C}^{t\text{-}\mathsf{cnf}}$ for $t = O(1)$ is adaptively pseudorandom on constrained points and Q-collusion resistant for any $Q = O(1)$, assuming adaptive pseudorandomness of the underlying PRF Π_{PRF}.*

Proof. We show the theorem by considering the following sequence of games between an adversary $\mathcal{A}$ against the pseudorandomness on constrained points security game and the challenger. In the following, for simplicity, we say an adversary $\mathcal{A}$ against the CPRF pseudorandomness game. Below, let E_i denote the probability that $\widehat{\mathsf{coin}} = \mathsf{coin}$ holds in Game_i. Recall that $\mathcal{A}$ makes at most Q-constrained key queries, where Q is a constant.

Game_0**:** This is defined as the ordinary CPRF pseudorandomness game played between $\mathcal{A}$ and the challenger. In particular, at the beginning of the game the challenger prepares the empty sets S_{eval} and S_{con}. In this game, the challenger responds to the queries made by $\mathcal{A}$ as follows:

- When $\mathcal{A}$ submits $x \in \{0,1\}^n$ as the evaluation query, the challenger returns $y \overset{\$}{\leftarrow} \mathsf{CPRF.Eval}(\mathsf{K}, x)$ to $\mathcal{A}$ and updates $S_{\mathsf{eval}} \leftarrow S_{\mathsf{eval}} \cup \{x\}$.
- When $\mathcal{A}$ submits $C^{t\text{-}\mathsf{cnf}^{(k)}} \in \mathcal{C}_n^{t\text{-}\mathsf{cnf}}$ as the k^{th} ($k \in [Q]$) constrained key query, the challenger returns $\mathsf{K}_{C^{(k)}} \overset{\$}{\leftarrow} \mathsf{CPRF.Constrain}(\mathsf{K}, C^{t\text{-}\mathsf{cnf}^{(k)}})$ to $\mathcal{A}$ and updates $S_{\mathsf{con}} \leftarrow S_{\mathsf{con}} \cup \{C^{t\text{-}\mathsf{cnf}^{(k)}}\}$.

Furthermore, recall that when $\mathcal{A}$ submits the target input $x^* \in \{0,1\}^n$ as the challenge query, we have the restriction $x^* \notin S_{\mathsf{eval}}$ and $x^* \in \mathsf{ConPoint}(C^{t\text{-}\mathsf{cnf}^{(k)}})$ for all $C^{t\text{-}\mathsf{cnf}^{(k)}} \in S_{\mathsf{con}}$. Here, the latter condition is equivalent to

$$\bigwedge_{J \in \mathcal{S}} C_J^{(k)}(x^*_J) = 0 \quad \text{for all} \quad C^{t\text{-}\mathsf{cnf}^{(k)}} \in S_{\mathsf{con}}, \tag{24}$$

where we express $C^{t\text{-}\mathsf{cnf}^{(k)}} := \{(J, C_J^{(k)})\}_{J \in \mathcal{S}}$. By definition, we have $|\Pr[\mathsf{E}_0] - 1/2| = \epsilon$.

Game_1**:** In this game, we add an extra abort condition for the challenger. Specifically, at the end of the game, the challenger samples a random element $\mathbb{J}^* = (J_1, \cdots, J_Q) \overset{\$}{\leftarrow} \mathcal{S}^Q$. The challenger further samples $\mathbb{W}^* = (w_1, \cdots, w_Q) \overset{\$}{\leftarrow} (\{0,1\}^t)^Q$ for all $k \in [Q]$. Then, the challenger checks whether the following equation holds with respect to the constrained key queries and the challenge query made by the adversary $\mathcal{A}$ at the end of the game:

- The challenger aborts if there exists $k \in [Q]$ such that
$$C^{t\text{-}\mathsf{cnf}^{(k)}}_{J_k}(w_k) = 1 \tag{25}$$
is satisfied.
- The challenger aborts if x^* does not satisfy
$$(\mathbb{W}^* \overset{?}{=} x^*_{\mathbb{J}^*}) = \bigwedge_{k \in [Q]} (w_k \overset{?}{=} x^*_{J_k}) = 1 \tag{26}$$
- The challenger aborts if $(\mathbb{J}^*, \mathbb{W}^*)$ chosen by the challenger does not equal to the *first pair* (with respect to some pre-defined order over $\mathcal{S}^Q \times (\{0,1\}^t)^Q$ such as the lexicographic order) that satisfies Equation (25) for all $k \in [Q]$ and Equation (26). Note that it is possible to efficiently find such a pair by enumerating over $\mathcal{S}^Q \times (\{0,1\}^t)^Q$ since $t, Q = O(1)$.[21]

When the challenger aborts, it substitutes the guess $\widehat{\mathsf{coin}}$ outputted by $\mathcal{A}$ with a random bit. We call this event abort.

[21] The reason why we require aborting the simulation is identical to that described in the proof of Theorem 4.2.

As we will show in Lemma B.4, there exists at least a single pair $(\mathbb{J}^*, \mathbb{W}^*) \in \mathcal{S}^Q \times (\{0,1\}^t)^Q$ that satisfies Equation (25) for all $k \in [Q]$ and Equation (26). Therefore, the event abort occurs with probability $1-1/(|\mathcal{S}|\cdot 2^t)^Q$ where $|\mathcal{S}| = \binom{n}{t}$. Furthermore, it can be seen that abort occurs independently from the view of $\mathcal{A}$. Therefore, we have

$$\begin{aligned} |\Pr[\mathsf{E}_1] - 1/2| &= |\Pr[\mathsf{E}_0] \cdot \Pr[\neg\mathsf{abort}] + (1/2) \cdot \Pr[\mathsf{abort}] - 1/2| \\ &= |\Pr[\mathsf{E}_0] \cdot (1/(|\mathcal{S}| \cdot 2^t)^Q) + (1/2) \cdot (1 - 1/(|\mathcal{S}| \cdot 2^t)^Q) - 1/2| \\ &= \epsilon/(|\mathcal{S}| \cdot 2^t)^Q, \end{aligned}$$

where we used the fact that $\widehat{\mathsf{coin}}$ is randomly chosen and thus equals to coin with probability $1/2$ when abort occurs. As in Remark B.2, if $t, Q = O(1)$, then $(|S| \cdot 2^t)^Q = \mathsf{poly}(\kappa)$.

Game_2**:** Recall that in the previous game, the challenger aborts at the end of the game, if the abort condition is satisfied. In this game, we change the game so that the challenger chooses $\mathbb{J}^*$ and $\mathbb{W}^*$ at the beginning of the game and aborts as soon as either $\mathcal{A}$ makes a constrained key query $C^{t\text{-}\mathsf{cnf}(k)} \in \mathcal{C}_n^{t\text{-}\mathsf{cnf}}$ that does not satisfy Equation (25) or a challenge query for x^* that does not satisfy Equation (26). Furthermore, it aborts if $(\mathbb{J}^*, \mathbb{W}^*)$ is not the first pair that satisfies Equation (25) for all $k \in [Q]$ and Equation (26). Since this is only a conceptual change, we have

$$\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_1].$$

Game_3**:** In this game, we change how the challenger responds to the challenge query when $\mathsf{coin} = 0$. For all the evaluation query and constrained key query, the challenger acts exactly the same way as in the previous game. In the previous game Game_2, when the adversary submits the target input $x^* \in \{0,1\}^n$ as the challenge query, the challenger first checks whether the condition in Equation (26) holds. If not, it aborts. Otherwise, it samples $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. In case $\mathsf{coin} = 0$, it computes $\mathsf{CPRF.Eval}(\mathsf{K}, x^*)$ as

$$y = \bigoplus_{\mathbb{J} \in \mathcal{S}^Q} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}, x_{\mathbb{J}}}, x), \tag{27}$$

using the master key

$$\mathsf{K} = \Big((\mathsf{K}_{\mathbb{J},\mathbb{W}}), (\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}}) \Big)_{\mathbb{J} \in \mathcal{S}^Q, \mathbb{W} \in (\{0,1\}^t)^Q}$$

that it constructed at the beginning of the game, where $\mathsf{K}_{\mathbb{J},\mathbb{W}}, \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ for all $\mathbb{J} \in \mathcal{S}^Q$ and $w \in (\{0,1\}^t)^Q$. Due to the condition in Equation (26), i.e., $\bigwedge_{k \in [Q]} \left(w_k \overset{?}{=} x^*_{J_k} \right) = 1$, we can rewrite Equation (27) as

$$y^* = \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}, x^*) \oplus \left(\bigoplus_{\mathbb{J} \in \mathcal{S}^Q \setminus \{\mathbb{J}^*\}} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}, x^*_{\mathbb{J}}}, x^*) \right). \tag{28}$$

In this game Game_3, when $\mathsf{coin} = 1$, the challenger instead samples a random $\bar{y}^* \xleftarrow{\$} \{0,1\}^k$ and returns the following to $\mathcal{A}$ instead of returning y^* to $\mathcal{A}$ as in Equation (28):

$$y^* = \bar{y}^* \oplus \left(\bigoplus_{\mathbb{J} \in \mathcal{S}^Q \setminus \{\mathbb{J}^*\}} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}, x^*_{\mathbb{J}}}, x^*) \right). \tag{29}$$

We show in Lemma B.5 that

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\kappa)$$

assuming pseudorandomness of the underlying PRF Π_{PRF}. In this game Game_3, the distribution of y^* for $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ are exactly the same since $\mathcal{A}$ has not made an evaluation query on x^* and $\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}$ is not given through any of the constrained key query. Concretely, $\bar{y}^*$ is distributed uniform random regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$ and thus the value of coin is information theoretically hidden to $\mathcal{A}$. Therefore, we have

$$\Pr[\mathsf{E}_3] = 1/2.$$

Combining everything together with Lemma B.4 and Lemma B.5, we have

$$\epsilon = |\Pr[\mathsf{E}_0] - 1/2| \leq (|\mathcal{S}| \cdot 2^t)^Q \cdot (|\Pr[\mathsf{E}_3] - 1/2| + \mathsf{negl}(\kappa)) = \mathsf{negl}(\kappa),$$

where the last equality follows by recalling that $n = \mathsf{poly}(\kappa)$, $|\mathcal{S}| = \binom{n}{t}$, and t and Q are constants.

Lemma B.4. *In* Game_1, *we have*

$$\left\{ (\mathbb{J}^*, \mathbb{W}^*) \in \mathcal{S}^Q \times (\{0,1\}^t)^Q \;\middle|\; \begin{array}{l} (\mathbb{J}^*, \mathbb{W}^*) \textit{ satisfies Equation (25)} \\ \textit{for all } k \in [Q]\textit{, and Equation (26)} \end{array} \right\} \neq \emptyset.$$

Proof. By the restriction posed on $\mathcal{A}$ in the game, for all $k \in [Q]$, there exists $J^{(k)} \in \mathcal{S}$ such that

$$C^{(k)}_{J^{(k)}}(x^*_{J^{(k)}}) = 0.$$

Let us denote $\bar{\mathbb{J}} := (J^{(1)}, \cdots, J^{(Q)}) \in \mathcal{S}^Q$ and $\bar{\mathbb{W}} := (x^*_{J^{(1)}}, \cdots, x^*_{J^{(Q)}}) \in (\{0,1\}^t)^Q$. It is easy to check that Equation (25) for all $k \in [Q]$ and Equation (26) hold if $\mathbb{J}^* = \bar{\mathbb{J}}$ and $\mathbb{W}^* = \bar{\mathbb{W}}$. □

Lemma B.5. *We have* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\kappa)$ *assuming that the underlying PRF* Π_{PRF} *satisfies adaptive pseudorandomness.*

Proof. For the sake of contradiction, let us assume an adversary $\mathcal{A}$ that distinguishes Game_2 and Game_3 with non-negligible probability ϵ'. We then construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of Π_{PRF} with the same probability. The adversary $\mathcal{B}$ proceeds as follows.

At the beginning of the game $\mathcal{B}$ samples a random tuple $\mathbb{J}^* = (J_1, \cdots, J_Q) \xleftarrow{\$} \mathcal{S}^Q$ and $\mathbb{W}^* = (w_1, \cdots, w_Q) \xleftarrow{\$} (\{0,1\}^t)^Q$ as in the Game_2-challenger. Then, it further samples $\mathsf{K}_{\mathbb{J},\mathbb{W}}, \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$ for all $\mathbb{J} \in \mathcal{S}^Q$ and $\mathbb{W} \in (\{0,1\}^t)^Q$ *except* for $\widehat{\mathsf{K}}_{\mathbb{J}^*,\mathbb{W}^*}$. It then sets the (simulated) master key K^* as

$$\mathsf{K}^* = \Big((\mathsf{K}_{\mathbb{J},\mathbb{W}})_{\mathbb{J} \in \mathcal{S}^Q, \mathbb{W} \in (\{0,1\}^t)^Q \setminus \{(\mathbb{J}^*,\mathbb{W}^*)\}}, (\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}})_{\mathbb{J} \in \mathcal{S}^Q, \mathbb{W} \in (\{0,1\}^t)^Q} \Big).$$

Here, $\mathcal{B}$ implicitly sets $\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}$ as the PRF key used by its PRF challenger. Finally, $\mathcal{B}$ prepares two empty sets $S_{\mathsf{eval}}, S_{\mathsf{con}}$. $\mathcal{B}$ then simulates the response to the queries made by $\mathcal{A}$ as follows:

- When $\mathcal{A}$ submits $x \in \{0,1\}^n$ as the evaluation query, $\mathcal{B}$ checks whether $x_{\mathbb{J}^*} = \mathbb{W}^*$. If not, then it can use the simulated master key K^* to compute

  $$y = \bigoplus_{\mathbb{J} \in \mathcal{S}^Q} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J},x_{\mathbb{J}}}, x).$$

  Otherwise, it makes an evaluation query to its PRF challenger on the input x. When it receives back $\bar{y}$ from the PRF challenger, $\mathcal{B}$ computes the output as

  $$y = \bar{y} \oplus \left(\bigoplus_{\mathbb{J} \in \mathcal{S}^Q \setminus \mathbb{J}^*} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J},x_{\mathbb{J}}}, x) \right).$$

  Finally, $\mathcal{B}$ returns y to $\mathcal{A}$ and updates $S_{\mathsf{eval}} \leftarrow S_{\mathsf{eval}} \cup \{x\}$. Note that by the specification of the PRF challenger, we have $\bar{y} = \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}, x)$.

- When $\mathcal{A}$ submits $C^{t\text{-}\mathsf{cnf}(k)} \in \mathcal{C}_n^{t\text{-}\mathsf{cnf}}$ as the k^{th} ($k \in [Q]$) constrained key query, $\mathcal{B}$ checks whether the condition in Equation (25) holds. If not it aborts and outputs a random bit. Otherwise, it returns the following constrained key $\mathsf{K}_{C^{(k)}}$ to $\mathcal{A}$:

$$\mathsf{K}_{C^{(k)}} = \left(\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}\right)_{\mathbb{J}\in\mathcal{S}^Q,\mathbb{W}\in(\{0,1\}^t)^Q},$$

  where $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}} = \mathsf{K}_{\mathbb{J},\mathbb{W}}$ if and only if $\mathbb{W} \in G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}(C^{t\text{-}\mathsf{cnf}(k)}, \mathbb{J}) = \{\mathbb{W} \in (\{0,1\}^t)^Q \mid \bigwedge_{i\in[Q]} C_{J_i}^{(k)}(w_i) = 1\}$ and $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}} = \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}}$ otherwise. Here, $\mathcal{B}$ can prepare all the PRF keys since the condition in Equation (25) guarantees us that we have $\mathbb{W}^* \notin G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}(C^{t\text{-}\mathsf{cnf}(k)}, \mathbb{J})$, or equivalently, $\bigwedge_{i\in[Q]} C_{J_i}^{(k)}(w_i) = 0$. Namely, $\widetilde{\mathsf{K}}_{\mathbb{J}^*,\mathbb{W}^*} = \widehat{\mathsf{K}}_{\mathbb{J}^*,\mathbb{W}^*}$ and so $\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}$ is not included in $\mathsf{K}_{C^{(k)}}$.

- When $\mathcal{A}$ submits the target input $x^* \in \{0,1\}^n$ as the challenge query, $\mathcal{B}$ checks whether the condition in Equation (26) holds. If not it aborts and outputs a random bit. Otherwise, $\mathcal{B}$ queries its PRF challenger on x^* as its challenge query and receives back $\bar{y}^*$. It then computes y^* as in Equation (29) and returns y^* to $\mathcal{A}$. Here, since Equation (26) holds, $\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}$ must be required to compute on input x^*.

Finally, $\mathcal{A}$ outputs its guess $\widehat{\mathsf{coin}}$. $\mathcal{B}$ then checks whether $(\mathbb{J}^*, \mathbb{W}^*)$ is the first pair that satisfies Equation (25) for all $k \in [Q]$ and Equation (26). If it does not hold, $\mathcal{B}$ outputs a random bit. Otherwise, $\mathcal{B}$ outputs $\widehat{\mathsf{coin}}$ as its guess.

This completes the description of $\mathcal{B}$. It is easy to check that in case $\mathsf{coin} = 0$, $\mathcal{B}$ receives $\bar{y}^* \xleftarrow{\$} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}^*,\mathbb{W}^*}, x^*)$, hence $\mathcal{B}$ simulates Game_2 perfectly. Otherwise in case $\mathsf{coin} = 1$, $\mathcal{B}$ receives $\bar{y}^* \xleftarrow{\$} \{0,1\}^k$, hence $\mathcal{B}$ simulates Game_3 perfectly. Therefore, we conclude that $\mathcal{B}$ wins the PRF pseudorandomness game with probability exactly ϵ'. Assuming that Π_{PRF} is pseudorandom, this is a contradiction, hence, ϵ' must be negligible. □

This completes the proof. □

Theorem B.6. *If the underlying PRF Π_{PRF} is adaptively pseudorandom, then our above CPRF Π_{CPRF} for the t-CNF predicate $\mathcal{C}^{t\text{-}\mathsf{cnf}}$ satisfies perfect weak 1-key privacy.*

Proof. Notice that the master key is of the form:

$$\left((\mathsf{K}_{\mathbb{J},\mathbb{W}}), (\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}})\right)_{\mathbb{J}\in\mathcal{S}^Q,\mathbb{W}\in(\{0,1\}^t)^Q}.$$

where $\mathsf{K}_{\mathbb{J},\mathbb{W}}, \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$. Let $C^{t\text{-}\mathsf{cnf}(0)}$ and $C^{t\text{-}\mathsf{cnf}(1)}$ be the two t-CNF predicates the adversary $\mathcal{A}$ queries. Then, $\mathcal{A}$ receives either one of the following two distributions:

- $\left(\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}^{(0)}\right)_{\mathbb{J}\in\mathcal{S}^Q,\mathbb{W}\in(\{0,1\}^t)^Q}$ where $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}^{(0)} = \mathsf{K}_{\mathbb{J},\mathbb{W}}$ if and only if $\mathbb{W} \in G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}(C^{t\text{-}\mathsf{cnf}(0)}, \mathbb{J})$, and $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}} = \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}}$ otherwise.
- $\left(\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}^{(1)}\right)_{\mathbb{J}\in\mathcal{S}^Q,\mathbb{W}\in(\{0,1\}^t)^Q}$ where $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}}^{(1)} = \mathsf{K}_{\mathbb{J},\mathbb{W}}$ if and only if $\mathbb{W} \in G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}(C^{t\text{-}\mathsf{cnf}(1)}, \mathbb{J})$, and $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{W}} = \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{W}}$ otherwise.

Notice that both the distributions are made up entirely of keys sampled from $\mathsf{PRF.Gen}$. Moreover, $\mathcal{A}$ cannot compare outputs under the constrained key and the real master key since $\mathcal{A}$ has no access to the evaluation oracle in this setting. Therefore, the two distributions are perfectly indistinguishable and the proof of weak key privacy is complete. □

C CPRF for Inner Product Implies CPRF for $O(1)$-CNF.

Here, we show that CPRFs for inner prouct predicates implies CPRFs for t-CNF predicates for a constant t. (See Definition B.1 for the definition of t-CNF predicates.) Let $\mathcal{C}$ be the set of all preficates $C : \{0,1\}^n \to \{0,1\}$ whose output depends at most t bits of the input. Then we have $|\mathcal{C}| = \binom{n}{t} \cdot 2^t = \mathsf{poly}(\kappa)$ for $n = \mathsf{poly}(\kappa)$ and $t = O(1)$. We index each element of $\mathcal{C}$ in an arbitrary way to denote $\mathcal{C} = \{C_1, C_2, ..., C_{|\mathcal{C}|}\}$. Then any t-CNF predicate

$C^{t\text{-cnf}}: \{0,1\}^n \to \{0,1\}$ can be written as $C^{t\text{-cnf}}(x) = \bigwedge_{i \in \mathcal{C}} C_i(x)$ for all $x \in \{0,1\}^n$. We let $T_{C^{t\text{-cnf}}}$ be the set of all $i \in [\ell]$ such that the term $C_i(x)$ appears in the above expression.

Let $\Pi_{\mathsf{CPRF}} = (\mathsf{CPRF.Setup}, \mathsf{CPRF.Gen}, \mathsf{CPRF.Eval}, \mathsf{CPRF.Constrain}, \mathsf{CPRF.ConstrainEval})$ be a CPRF for inner products over $\mathcal{D} := [-B, B]^\ell \subset \mathbb{Z}^\ell$ for $\ell := |\mathcal{C}| + 1$ and $B \geq |\mathcal{C}|$ as constructed in Section 5. Then we construct a CPRF scheme $\Pi'_{\mathsf{CPRF}} = (\mathsf{CPRF.Setup}', \mathsf{CPRF.Gen}', \mathsf{CPRF.Eval}', \mathsf{CPRF.Constrain}', \mathsf{CPRF.ConstrainEval}')$ for t-CNF predicates as follows:

$\mathsf{CPRF.Setup}'(1^\kappa)$**:** It just runs $\mathsf{CPRF.Setup}(1^\kappa)$.

$\mathsf{CPRF.Gen}'(\mathsf{pp})$**:** It just runs $\mathsf{CPRF.Gen}(\mathsf{pp})$.

$\mathsf{CPRF.Eval}'(\mathsf{K}, x)$**:** On input the master secret key K and $x \in \{0,1\}^n$, it computes $\mathbf{x} := (C_1(x), C_2(x), ..., C_{|\mathcal{C}|}(x), -1)$ and outputs $\mathsf{CPRF.Eval}(\mathsf{K}, \mathbf{x})$.

$\mathsf{CPRF.Constrain}'(\mathsf{K}, C^{t\text{-cnf}})$**:** On input the master secret key K and a t-CNF $C^{t\text{-cnf}}$, it computes $\mathbf{y} := (y_1, ..., y_\ell, |T_{C^{t\text{-cnf}}}|)^\top$ where $y_i = 1$ if $i \in T_{C^{t\text{-cnf}}}$ and $y_i = 0$ otherwise, and outputs $\mathsf{CPRF.Constrain}(\mathsf{K}, \mathbf{y})$.

$\mathsf{CPRF.ConstrainEval}'(\mathsf{K}_{C^{t\text{-cnf}}}, x)$**:** On input a constrained key $\mathsf{K}_{C^{t\text{-cnf}}}$ and $x \in \{0,1\}^n$ it computes $\mathbf{x} := (C_1(x), C_2(x), ..., C_{|\mathcal{C}|}(x), -1)$ and outputs $\mathsf{CPRF.ConstrainEval}(\mathsf{K}_{C^{t\text{-cnf}}}, \mathbf{x})$.

This completes the construction. It is easy to see that for any t-CNF $C^{t\text{-cnf}}$ and $x \in \{0,1\}^n$, we have

$$C^{t\text{-cnf}}(x) = 1 \iff \langle \mathbf{x}, \mathbf{y} \rangle = 0$$

where $\mathbf{x}$ and $\mathbf{y}$ are the corresponding vectors to $C^{t\text{-cnf}}$ and x as described in the construction. Therefore correctness and security are inherited from Π_{CPRF} to Π'_{CPRF}.

D Proof of Lemma 6.3

Here, we prove Lemma 6.3. Suppose that a CPRF scheme $\Pi_{\mathsf{CPRF}} = (\mathsf{CPRF.Setup}, \mathsf{CPRF.Gen}, \mathsf{CPRF.Eval}, \mathsf{CPRF.Constrain}, \mathsf{CPRF.ConstrainEval})$ 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 we prove that the scheme is adaptively secure against 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. For the sake of contradiction, we assume that there exists an adversary $\mathcal{A}$ that makes at most $Q_1 - 1$ pre-challenge constrained key queries, Q_2 post-challenge constrained key queries, and $Q_{\mathsf{eval}} = \mathsf{poly}(\kappa)$ evaluation queries and breaks the adaptive security of Π_{CPRF} with a non-negligible advantage $\epsilon(\kappa)$. We define a noticeable function $\epsilon_0(\kappa)$ such that $\epsilon(\kappa) \geq \epsilon_0(\kappa)$ for infinitely many κ. (Such ϵ_0 must exist by the definitions of non-negligible and noticeable functions.) Let P_K, PrtSmp and $\mathsf{H}_{\mathsf{adm}}$ be as defined in Definition 3.6. Then we construct an adversay $\mathcal{A}'$ that makes at most Q_1 pre-challenge constrained key queries, Q_2 post-challenge constrained key queries, and no evaluation queries and breaks the adaptive security of Π_{CPRF} as follows:

$\mathcal{A}'(\mathsf{pp})$**:** Given the public parameter pp, $\mathcal{A}'$ picks $K \overset{\$}{\leftarrow} \mathsf{PrtSmp}(1^\kappa, Q_{\mathsf{eval}}, \epsilon_0)$, and queries a circuit $C[K]$ that computes $P_K(\mathsf{H}_{\mathsf{adm}}(\cdot))$ as a constrained key query, and obtains a corresponding constrained key $\mathsf{K}_{C[K]}$. Then it runs $\mathcal{A}$ on input pp. When $\mathcal{A}$ makes a constrained key query C, $\mathcal{A}'$ just forwards the query to its own constrained key oracle to obtain a constrained key K_C, and returns K_C to $\mathcal{A}$. When $\mathcal{A}$ makes an evaluation query x, $\mathcal{A}'$ immediately aborts and outputs a random bit if $P_K(\mathsf{H}_{\mathsf{adm}}(x)) = 0$. Otherwise, $\mathcal{A}'$ computes $y \leftarrow \mathsf{CPRF.ConstrainEval}(\mathsf{pp}, \mathsf{K}_{C[K]}, x)$ and returns y to $\mathcal{A}$. When $\mathcal{A}$ makes a challenge query x^*, $\mathcal{A}'$ immediately aborts and outputs a random bit if $P_K(\mathsf{H}_{\mathsf{adm}}(x^*)) = 1$. Otherwise, $\mathcal{A}'$ queries x^* as its own challenge query to obtain a challenge value y^*, and returns y^* to $\mathcal{A}$. Finally, when $\mathcal{A}$ outputs its guess $\widehat{\mathsf{coin}}$, $\mathcal{A}'$ outputs the same bit $\widehat{\mathsf{coin}}$ (if it has not aborted).

This completes the description of $\mathcal{A}'$. It is clear that $\mathcal{A}'$ makes no evaluation query and one extra pre-challenge constrained key query in addition to ones made by $\mathcal{A}$. We remark that $\mathcal{A}'$ is a valid adversary since we have $P_K(\mathsf{H}_{\mathsf{adm}}(x^*)) = 0$ as long as it does not abort and $C(x^*) = 0$ for all constrained key queries C made by $\mathcal{A}$ since $\mathcal{A}$ is assumed to be a valid adversary. Let $\gamma_{\min}(\kappa)$ and $\gamma_{\max}(\kappa)$ be as defined in Definition 3.6. Then the probability that $\mathcal{A}'$ does not abort is between $\gamma_{\min}(\kappa)$ and $\gamma_{\max}(\kappa)$. Applying Lemma 3.10, we have

$$|\Pr[\mathcal{A}' \text{ wins}] - 1/2| \geq \gamma_{\min} \cdot \epsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2} \geq \gamma_{\min} \cdot \epsilon_0 - \frac{\gamma_{\max} - \gamma_{\min}}{2}$$

for infinitely many κ. Since $\gamma_{\min} \cdot \epsilon_0 - \frac{\gamma_{\max} - \gamma_{\min}}{2}$ is noticeable by Definition 3.6, $\mathcal{A}'$ breaks the adaptive security of Π_{CPRF} with non-negligible advantage. This completes the proof of Lemma 6.3.

Contents