Constrained PRFs for Bit-fixing (and More) from OWFs with Adaptive Security and Constant Collusion Resistance[*]

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

[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@aist.go.jp, yamada-shota@aist.go.jp
[3]NTT Secure Platform Laboratories, Tokyo, Japan
ryo.nishimaki.zk@hco.ntt.co.jp

June 3, 2019

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 sort of predicate. First introduced by Boneh and Waters [AC'13], Kiayias et al. [CCS'13] and Boyle et al. [PKC'14], they have been shown to be a useful cryptographic primitive with many applications. The full security definition of CPRFs requires the adversary to learn multiple constrained keys in an arbitrary order, a requirement for many of these applications. Unfortunately, existing constructions of CPRFs satisfying this security notion are only known from exceptionally strong cryptographic assumptions, such as indistinguishability obfuscation (IO) and the existence of multilinear maps, even for very weak constraints. CPRFs from more standard assumptions only satisfy selective security for a single constrained key query.

In this work, we give the first construction of a CPRF that can adaptively issue a constant number of constrained keys for bit-fixing predicates (or more generally t-conjunctive normal form predicates), only requiring the existence of one-way functions (OWFs). This is a much weaker assumption compared with all previous constructions. In addition, we prove that the new scheme satisfies 1-key privacy (otherwise known as constraint-hiding). This is the only construction for any non-trivial predicates to achieve adaptive security and collusion-resistance outside of the random oracle model or relying on strong cryptographic assumptions. Our technique represents a noted departure from existing CPRF constructions.

1 Introduction

Historically, pseudorandom functions (PRFs) provide the basis of a huge swathe of cryptography. Intuitively, such a function takes a uniform key and some binary string x as input, and outputs (deterministically) some value y. The pseudorandomness of the function dictates that y is indistinguishable from the output of a uniformly sampled function operating solely on x. PRFs typically provide useful sources of randomness in cryptographic constructions that take adversarially-chosen inputs. Simple constructions of PRFs exist based on well-known standard assumptions: Goldreich, Goldwasser, and Micali give a

[*]This work is a merged version of [DN18] and [KY18] with additional results.
[†]This work was done part in while the author undertook a research internship at NTT.

construction based on the existence of pseudorandom generators [GGM86]; Naor and Reingold give a simple construction from assumptions related to the discrete log problem [NR04].

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 that can evaluate the function on a subset of the input space defined by a function predicate. Specifically, keys can be constrained with respect to predicates $C \in \mathcal{C}$ where $\mathcal{C}$ is some specific function class of constraints. Using this formulation, the value $\mathsf{CPRF.Eval}(\mathsf{K}, x)$ can only be evaluated using a constraint key for C if $C(x) = 1$. If $C(x) = 0$ then the result of the evaluation is no longer defined (or correctness is lost).

There are various security requirements for CPRFs considered in the literature. One of the most natural and well-studied ones is 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 $C_i \in \mathcal{C}$ and $i \in [Q]$ for $Q = \mathsf{poly}$.[1] The security requirement dictates that the CPRF remains pseudorandom on an input point $x^\dagger$, where $C_i(x^\dagger) = 0$ for all i. Here, we differentiate the strength of the security notion in two measures: when the adversary can query the constrained keys in an arbitrary manner, then we say the CPRF is *adaptively* secure on constrained points, and when $Q > 1$, then we say the CPRF is *collusion-resistant*.

In this work, our main question is:

Can we construct constrained PRFs that are adaptively secure on constrained points with collusion-resistance, for expressive classes of predicates based on standard assumptions?

Above, the notion of an expressive predicate is largely subjective, the most expressive predicates that are considered for CPRFs are based around circuit-based predicates of varying depths. Previous works have managed to achieve CPRF functionality for expressive circuit-based predicates, but without achieving adaptive security and collusion-resistance [BV15, CC17, BTVW17, CVW18, PS18] (i.e., only permitting one constraint key query before the public parameters are generated by the challenger).[2] CPRFs for circuit-based predicates that do achieve adaptive security and/or collusion-resistance require making assumptions over the existence of cryptographically blunt instruments such as multilinear maps, indistinguishability obfuscation (IO) and random oracles (ROM) [BW13, BLW17, HKKW14]. The most expressive predicates that have been achieved in CPRF constructions — without requiring strong assumptions or the ROM — that attain adaptive security and collusion-resistance are only limited to predicates that match bit-strings x against some specified prefix [BFP⁺15].

Thus, in this work, we will consider classes of '*expressive predicates*' to be those that provide more than the prefix functionality achieved previously. We will also consider whether it is possible to attain stronger security properties such as privately constrained keys [BLW17]. We will discuss and compare the existing CPRF schemes in more detail in Section 1.1.

PREDICATES. We briefly review the predicates that have been considered in the literature 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 listed in ascending order of expressibility:[3]

- puncturing: $C_v(x) = 1$ iff $x \neq v$;
- 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 $v^b \in \{0,1\}^l$;

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

[2] If more than one query is permitted, then all of the constructions admit attacks that result in a complete loss of security.

[3] To be precise, we note that the class of prefix-fixing is not necessarily more expressive than puncturing. In particular, they are incomparable.

- 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 $C_i \in \mathsf{NC}^0_t$ where NC^0_t is the class of NC^0 circuits that read at most t indices of the input;
- general circuits in $\{\mathsf{NC}^1, \mathsf{P/poly}\}$: $C(x) = 1$ where $C \in \{\mathsf{NC}^1, \mathsf{P/poly}\}$.

In this work the predicates that we will be considering are the bit-fixing predicates and the t-conjunctive normal form (t-CNF) 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. Circuit-based predicates are the only class of predicates more expressive than t-CNF predicates that have been considered in the literature.

1.1 Existing Constructions

Since the original works of [BW13, KPTZ13, BGI14], numerous constructions of CPRFs have been given, relying on different primitives and providing a range of functionality. We summarise, to the best of our knowledge, all known constructions in Table 1. 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, BGI13]. 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.

CPRFs supporting more flexible predicates such as left-right-fixing, bit-fixing, and $\mathsf{P/poly}$ circuit predicates were also considered in the original works of [BW13]. They showed such constructions in the random oracle model (ROM) or by assuming the existence of multilinear maps. With the help of the ROM or strong assumptions, these CPRFs for flexible predicates satisfy collusion-resistance for any polynomial number of constrained keys. Moreover, CPRFs for the left-right-fixing predicate satisfy adaptive-security in the ROM.

Recently, constructions of CPRFs for flexible predicates from much weaker assumptions have been considered, at the expense of providing weaker guarantees. The CPRF schemes of [BV15, CC17, BTVW17, PS18, CVW18] derive security from the learning with errors (LWE) assumption, and other lattice-based assumptions. All of these CPRFs allow for (the powerful) circuit-based constraints for either the NC^1 or $\mathsf{P/poly}$ classes. However, all of these constructions are only selectively secure 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 non-trivial predicates) or collusion resistance (even for 2 keys!).

1.1.1 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.[4] The CPRF for the prefix-fixing predicates based on the GGM-PRF [BW13, KPTZ13, BGI13] trivially achieves 1-key privacy. The constructions of [BLW17]

[4]They are also known as ‘constraint-hiding’ CPRFs.

satisfy poly-key privacy (hence poly constrained key queries) for circuit predicates under the existence of IO. 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 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.

1.2 Our Contribution

In this work, we develop a new CPRF construction for the t-CNF predicate, which in particular includes the bit-fixing predicate as a special case. While this predicate is less expressive than general bounded-depth circuit predicates, our construction is derived only from the existence of one-way functions; which is a remarkably weaker assumption than all other CPRF constructions for the bit-fixing predicate [BW13, BLW17, CC17, AMN$^+$18].

Our construction is the first to satisfy *adaptive* security or *collusion-resistance* for bit-fixing from any standard assumption and within the standard model. Specifically, our construction is secure against PPT adversaries who learn $Q = O(1)$ constrained keys in an arbitrary order (i.e., a constant number with respect to the security parameter). Our construction also provides a simple solution to an adaptively secure CPRF for prefix-fixing predicates where prior works incurred at least a quasi-polynomial reductions loss [BW13, KPTZ13, BGI13, FKPR14, JKK$^+$17] since prefix-fixing predicates are are a special case of bit-fixing predicates.

Finally, our construction satisfies (weak) 1-key privacy by the definition of [BLW17] (see Remark 3.6 for more details on the definition of key privacy). We are unable to achieve security for the setting where $m > 1$ and we leave this open as an interesting future research direction. We summarize our contribution alongside the previous state-of-the-art in Table 1.

Applications. The CPRF construction that we describe can be used as a building block in realising adaptively-secure t-CNF attribute-based-encryption (ABE) based on lattices, as 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 our bit-fixing PRF is not only a theoretical interest but also a useful tool to achieve higher security of advanced encryption.

2 Technical Overview

The original starting point for this work was an attempt to modify the bit-fixing CPRF of Canetti and Chen [CC17] (CC17) so that it achieved collusion-resistance. The CC17 construction was the first to make explicit use of a *directed line* evaluation procedure. At each node i there is a choice between two matrices $\boldsymbol{D}_i^{(b)}$ for $b \in \{0,1\}$, and there is also a public matrix $\boldsymbol{A}$. Evaluating the CPRF amounts to computing the rounded product $\boldsymbol{Y} \leftarrow \lfloor \boldsymbol{A} \prod_{i=1}^{\ell} \boldsymbol{D}_i^{(x_i)} \rceil$ for some input $x \in \{0,1\}^\ell$.[5] See Figure 1 for a diagrammatic representation of this procedure.

For constraining their CPRF with respect to some wildcard string $v \in \{0,1,*\}^\ell$, the user receives $\boldsymbol{D}_i^{(v_i)}$ for each $v_i \in \{0,1\}$, and $\boldsymbol{D}_i^{(b)}$ ($b \in \{0,1\}$) for $v_i = *$. Constrained evaluation is clearly then possible for any input x that satisfies the bit-fixing predicate specified by v.

The security of the scheme is derived from the fact that any constrained point $x^\dagger$ must satisfy $(x_i^\dagger \neq v_i) \wedge (v_i \neq *)$ for some $i \in [\ell]$. Then the evaluation on such a point comprises the product above, except where $\boldsymbol{D}_i^{(x_i^\dagger)}$ is unknown to the adversary. In the evaluation, this allows the simulator to argue

[5]The rounding is used to ensure deterministic output for the PRF.

Table 1: List of existing constructions of CPRFs along with their functionality and the assumptions required. In column 'Predicate', LR stands for left-right-fixing predicates, BF stands for bit-fixing predicates, and t-CNF stands for t-conjunctive normal form predicates. In column 'Assumption', BDDH stands for bilinear decisional Diffie-Hellman assumption, MDDH stands for multilinear decisional Diffie-Hellman assumption, LWE stands for learning with errors assumption, SGH stands for subgroup hiding assumption, and L-DDHI stands for L-decisional Diffie-Hellman inversion assumption. We do not consider the CPRFs of [Bit17, GHKW17] since they do not permit evaluation queries. (†) We note [BW13] and [BGI14] did not originally consider key-privacy.

	Adaptive	Collusion-resistance	Privacy	Predicate	Assumption
[BW13]	$\times$	1	$1^\dagger$	Prefix	OWF
	✓	poly	poly	LR	BDDH & ROM
	$\times$	poly	0	BF	MDDH
	$\times$	poly	0	P/poly	MDDH
[KPTZ13]	$\times$	1	1	Prefix	OWF
[BGI14]	$\times$	1	$1^\dagger$	Prefix	OWF
[HKKW14]	✓	poly	0	P/poly	IO & ROM
[BFP$^+$15]	$\times$	poly	0	Prefix	LWE
[BV15]	$\times$	1	0	P/poly	LWE
[HKW15]	✓	poly	0	Puncturing	SGH & IO
[BLW17]	$\times$	poly	1	Puncturing	MDDH
	$\times$	poly	1	BF	MDDH
	$\times$	poly	poly	P/poly	IO
[BTVW17]	$\times$	1	1	P/poly	LWE
[CC17]	$\times$	1	1	BF	LWE
	$\times$	1	1	NC^1	LWE
[AMN$^+$18]	$\times$	1	1	BF	DDH
	$\times$	1	0	NC^1	L-DDHI
	✓	1	1	BF	ROM
	✓	1	0	NC^1	L-DDHI & ROM
[CVW18]	$\times$	1	1	NC^1	LWE
[PS18]	$\times$	1	1	P/poly	LWE
[AMN$^+$19]	✓	1	0	NC^1	SGH & IO
This work	✓	$O(1)$	1	BF	OWF
	✓	$O(1)$	1	t-CNF	OWF

$$\begin{tikzcd}
\boxed{\boldsymbol{D}_1^{(0)}} & \boldsymbol{D}_2^{(0)} & \boldsymbol{D}_3^{(0)} & & \boxed{\boldsymbol{D}_\ell^{(0)}} \\
\bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \cdots \arrow[r] & \bullet \\
\boldsymbol{D}_1^{(1)} & \boxed{\boldsymbol{D}_2^{(1)}} & \boxed{\boldsymbol{D}_3^{(1)}} & & \boldsymbol{D}_\ell^{(1)}
\end{tikzcd}$$

Figure 1: Directed line representation used in CC17. The choices of matrices correspond to some input $x = 011\ldots0$. The CPRF is computed by multiplying each of these matrices as well as by the public matrix $\boldsymbol{A}$ on the LHS.

using the learning with errors (LWE) assumption with a public uniform matrix on the LHS, and the small secret matrix $\boldsymbol{D}_i^{(x_i^\dagger)}$, that the entire product is distributed uniformly randomly [BLMR13].

Allowing > 1 constrained key query. The difficulty with allowing more than one constrained key query is that all of the matrices in CC17 can be revealed to the adversary, while there are still numbers of points constrained. When all of the matrices are uncovered, then the LWE assumption can no longer be used to prove security since there are no private elements utilised in the product. As an example of how all such matrices can be uncovered. Consider the two wildcard string $v = 0 * * \ldots * *0$ and $\bar{v} = 1 * * \ldots * *1$. Notice that any binary string x of the form $x = 0 \ldots 1$ or $x = 1 \ldots 0$ does not satisfy the predicate and so such an input string would be constrained with respect to constrained keys $v, \bar{v}$. However, all matrices would also have to be revealed to the adversary to be able to evaluate the CPRF using the constrained keys for $v, \bar{v}$. As such, it is impossible to argue that the pseudorandomness of the CPRF holds in such a setting.

2.1 Combinatorial Techniques for Collusion-Resistance

In this work, we observe two ways of redefining the directed line evaluation of CC17 that result in concrete improvements in the scheme. For ease of presentation, we consider our CPRF for bit-fixing predicates in the technical overview rather than the more general CPRF for t-CNF predicates. The high-level idea is very similar and generalizes naturally.

Firstly, notice that the technique that CC17 uses is fundamentally independent of the LWE assumption. In essence, their security proof requires that for a given node, a hidden element (not revealed to the adversary) can be used to evaluate some intermediate pseudorandom output; each of these intermediate outputs is then combined into a single output for the entire PRF. The LWE assumption naturally gives methods for generating such pseudorandom samples. Secondly, it is clear that CC17 only allows one constrained key query because each input bit of the evaluation/wildcard predicate is analysed individually.

Removing the LWE assumption. We can remove the LWE assumption (and replace it with a much weaker assumption) by generating pseudorandom samples using different primitives. We reconfigure the CC17 construction such that there is an underlying pseudorandom function PRF, where $\mathsf{PRF.Eval} : \{0,1\}^\kappa \times \{0,1\}^\ell \mapsto \{0,1\}^n$, and then replace each of the matrices with uniformly sampled 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:

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

A modified depiction of the construction is given in Figure 2.

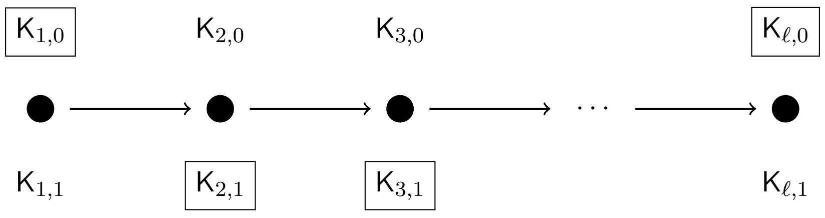

Figure 2: Modified directed line representation with keys for PRF representing each node, instead of small LWE matrices as in CC17. Here, the choices of PRF keys correspond to some input $x = 011 \cdots 0$.

The constraining algorithm for a wildcard string predicate $v \in \{0,1,*\}^\ell$ reveals individual keys K_{i,v_i} where $v_i \in \{0,1\}$ and pairs $\mathsf{K}_{i,b}$ ($b \in \{0,1\}$) where $v_i = *$. Constrained evaluation follows in the natural way defined by CC17, except using the underlying keys that are revealed to the user.

The security of the scheme still rests upon the fact that for a single constrained key, with respect to v, then for a constrained input $x^\dagger$ there exists a $j \in [\ell]$ such that $(x^\dagger_j \neq v_j) \wedge (v_j \neq *)$ is satisfied. The pseudorandomness of $y \leftarrow \mathsf{CPRF.Eval}(\mathsf{K}, x^\dagger)$ is achieved because

$$y \leftarrow \bigoplus_{i=1}^{\ell} \mathsf{PRF.Eval}(\mathsf{K}_{i,x^\dagger_i}, x^\dagger) = \mathsf{PRF.Eval}(\mathsf{K}_{j,x^\dagger_j}, x^\dagger) \oplus \left(\bigoplus_{i \neq j} \mathsf{PRF.Eval}(\mathsf{K}_{i,b}, x^\dagger) \right)$$

where $\mathsf{PRF.Eval}(\mathsf{K}_{j,x^\dagger_j}, x^\dagger)$ is a PRF evaluation using a key that is unknown to the adversary. As such, it is possible to replace this evaluation with a uniformly sampled $y_j \in \{0,1\}^n$ by the pseudorandomness of PRF. In turn, this results in a uniformly distributed CPRF output y and so pseudorandomness is ensured. It is possible to instantiate pseudorandom functions using only one-way functions [GGM86, HILL99].

Collusion-resistance for two constrained key queries. It should be clear that, while we may have achieved a CPRF from a much weaker assumption than the assumptions used in CC17, we have still not achieved collusion-resistance. As we mentioned in the case of CC17, the reason that more than one constrained key query is not permitted is that it would reveal all of the underlying PRF keys, while there would still be constrained inputs. The reason for this is that we examine each of the input bits individually for choosing the underlying keys, and then evaluate the PRF accordingly.

Consider a scheme that instead of considering one input bit at each node in Figure 2, instead it considered two input bits. For example, a modified depiction of the construction is given in Figure 3. In the set-up shown in Figure 3 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.

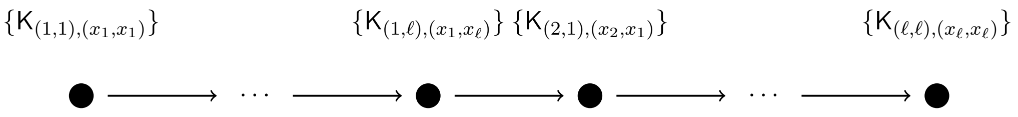
$\{\mathsf{K}_{(1,1),(x_1,x_1)}\}$
$\{\mathsf{K}_{(1,\ell),(x_1,x_\ell)}\}$
$\{\mathsf{K}_{(2,1),(x_2,x_1)}\}$
$\{\mathsf{K}_{(\ell,\ell),(x_\ell,x_\ell)}\}$

Figure 3: Directed line considering two input bits at each node, 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 3:

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

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 $v, \bar{v} \in \{0,1,*\}^\ell$. For an input $x^\dagger$ to be constrained with respect to both $v, \bar{v}$, then there exists an $i' \in [\ell]$ where $(x^\dagger_{i'} \neq v_{i'}) \wedge (v_{i'} \neq *)$ and likewise $(x^\dagger_{j'} \neq \bar{v}_{j'}) \wedge (\bar{v}_{j'} \neq *)$ for some $j' \in [\ell]$. Equivalently, we must have $x^\dagger_{i'} = 1 - v_{i'}$ and $x^\dagger_{j'} = 1 - \bar{v}_{j'}$ 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 not have been revealed to the adversary. Moreover, such a pair (i',j') exists for all constrained input queries $x^\dagger$.

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 $x^\dagger$ as:

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

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

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

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

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

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

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

For any set of Q constrained key queries associated with strings $v^{(1)}, \ldots, v^{(Q)}$ and any constrained input $x^\dagger$, there must be a vector $(i'_1, \ldots, i'_Q)$ such that $(x^\dagger_{i'_j} \neq v^{(j)}_{i'_j}) \wedge (v^{(j)}_{i'_j} \neq *)$ for all $j \in [Q]$. Therefore, the key $\mathsf{K}_{(i'_1,\ldots,i'_Q),(x^\dagger_{i'_1},\ldots,x^\dagger_{i'_Q})}$ is never revealed to the adversary. Finally, we can prove the pseudorandomness of the CPRF on input $x^\dagger$ 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 $(i_1, \ldots, i_Q)$, there are 2^Q underlying PRF keys associated with this node; moreover, there are ℓ^Q such nodes. Therefore the total size of K is $(2\ell)^Q$. As a result, considering $\ell = \mathsf{poly}$ since this is the input length of PRF, we are only able to afford $Q = O(1)$. This bound is inherent in the directed line paradigm because our technique is purely combinatorial.

2.2 Achieved Security Properties

Finally, we assess the security properties achieved by our CPRF. In particular, we observe that our construction satisfies adaptive security when the underlying pseudorandom functions satisfy adaptive pseudorandomness. Moreover, we observe that the constrained keys are constraint-hiding.

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 Table 1 for an overview. The sub-exponential security loss is incurred as previous constructions achieve adaptive security by letting the reduction guess the challenge point $x^\dagger$ that the adversary uses. 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. $K_{T^\dagger, x^\dagger_T}$ for $T^\dagger \subset [\ell], |T^\dagger| = Q$). 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, due to the non-trivial abort condition, there is a subtle technical issue we must resolve which is addressed in Lemma 4.3. Similar problems were identified by Waters [Wat05] who introduced the 'artificial abort step'.

1-KEY PRIVACY. The construction of CC17 achieves 1-key privacy by adding a set of dummy matrices $\bar{\boldsymbol{D}}_i^{(b)}$ to the master secret for each $i \in [\ell]$ and $b \in \{0, 1\}$. These matrices are distributed similarly to the non-dummy (functional) matrices. When revealing the constrained key to the adversary, for any matrix $\boldsymbol{D}_i^{(1-v_i)}$ that is not revealed, instead the matrix $\bar{\boldsymbol{D}}_i^{(1-v_i)}$ is revealed. Since the functional and non-dummy matrices are distributed similarly, the constrained key that the adversary learns is always distributed in the same way. Therefore the constrained key hides the constraint that is encoded.

For our construction, a similar philosophy follows by introducing dummy PRF keys that are sampled uniformly. Then, K contains only uniformly distributed keys from $\{0, 1\}^\kappa$, and so the constraint is perfectly hidden. Recall the indistinguishability security game for weak key privacy of Boneh et al. [BLW17]. In this game, the adversary chooses two bit-fixing predicates $(v^{(0)}, v^{(1)})$ and the challenger flips a bit b and returns $\mathsf{K}_{v^{(b)}}$. The adversary has to distinguish which constraint has been encoded into the key. Since both keys are simply made up of uniformly sampled PRF keys, then the resulting CPRF key is *perfectly* indistinguishable for either value of $b \in \{0, 1\}$. See the proof of Theorem 4.5 for more details.

We cannot obtain key privacy for $m > 1$ queries because 2 constrained keys would reveal where the constrained keys differ since the underlying functional keys have to be consistent across constrained keys. Therefore, receiving more than one constrained key, it will become obvious which keys correspond to which constraint. For a similar reason, we cannot obtain stronger simulation-based security [CC17] for our CPRF. In this model, Q-key privacy and Q-collusion-resistance are considered in parallel, and this would break the 1-key privacy of our scheme.

3 Preliminaries

3.1 Pseudorandom Functions

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

Syntax. Let $\ell = \ell(\kappa)$, and $n = n(\kappa)$ 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 $x \in \{0, 1\}^\ell$ and outputs $y \in \{0, 1\}^n$.

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 A and a challenger:

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

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

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

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

Guess: Eventually, A outputs b' as a guess for b.

We say the adversary A wins the game if $b' = b$.

Definition 3.1. *A PRF Π_{PRF} is said to be (adaptive) pseudorandom if for all PPT adversary A, the probability of 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].

3.2 Constrained Pseudorandom Functions

We now define constrained pseudorandom functions (CPRFs).

Syntax. Let $\ell = \ell(\kappa)$, and $n = n(\kappa)$ be integer-valued positive polynomials of the security parameter κ. 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 $\{0,1\}^\ell$ and range $\{0,1\}$ whose sizes are polynomially bounded. In the following we drop the subscript for clarity.

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

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

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

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

$\mathsf{CPRF.ConstrainEval}(\mathsf{K}_C, x) :\to y$: The constrained evaluation algorithm takes as input the constrained key K_C and an input $x \in \{0,1\}^\ell$ and outputs $y \in \{0,1\}^n$x.

Correctness. We define the notion of *correctness* for CPRFs. We say a CPRF Π_{CPRF} is correct if for all $\kappa \in \mathbb{N}$, $\ell, n \in \mathsf{poly}(\kappa)$, $\mathsf{K} \in \mathsf{CPRF.Gen}(1^\kappa)$, $C \in \mathcal{C}_\kappa$, $\mathsf{K}_C \in \mathsf{CPRF.Constrain}(\mathsf{K}, C)$, $x \in \{0,1\}^\ell$ such that $C(x) = 1$, we have $\mathsf{CPRF.Eval}(\mathsf{K}, x) = \mathsf{CPRF.ConstrainEval}(\mathsf{K}_C, x)$.

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 $C : \{0,1\}^\ell \to \{0,1\}^n$, let $\mathsf{ConPoint} : \mathcal{C} \to \{0,1\}^\ell$ be a function which outputs the set of all constrained points $\{x \mid C(x) = 0\}$. Here $\mathsf{ConPoint}$ is not necessarily required to be efficiently computable.

Formally, this security notion is defined by the following game between an adversary A and a challenger:

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

Queries: During the game, A can adaptively make the following two types of queries:

-**Evaluation Queries**: Upon a query $x \in \{0,1\}^\ell$, the challenger evaluates $y \leftarrow \mathsf{CPRF.Eval}(\mathsf{K}, x)$ and returns $y \in \{0,1\}^n$ to 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 $\mathsf{K}_C \leftarrow \mathsf{CPRF.Constrain}(\mathsf{K}, C)$ and returns K_C to A. It then updates $S_{\mathsf{con}} \leftarrow S_{\mathsf{con}} \cup \{C\}$.

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

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

Guess: Eventually, A outputs b' as a guess for b.

We say the adversary A wins the game if $b' = b$.

Definition 3.2. *A CPRF Π_{CPRF} is said to be (adaptive) pseudorandom on constrained points if for all PPT adversary A, $|\Pr[\mathsf{A}\ wins] - 1/2| = \mathsf{negl}(\kappa)$ holds.*

Remark 3.3 (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 3.4 (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 master key $\mathsf{K} \leftarrow \mathsf{CPRF.Gen}(1^\kappa)$.

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

[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.

Guess: A outputs b' as a guess for b.

We say the adversary A wins $b' = b$.

Definition 3.5. *A CPRF Π_{CPRF} is said to satisfy perfect weak 1-key privacy if for all PPT adversaries A, then $|\Pr[\mathsf{A}\ wins] - 1/2| = 0$ holds.*

Remark 3.6. 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 adaptive 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 $Q > 1$ in the standard model, excluding the CPRF for the trivial singleton sets $F = \{\{x\} \mid x \in \{0,1\}^\ell\}$ [BW13] or the selectively-secure and collusion-resistant CPRF for prefix-fixing predicates by [BFP$^+$15].

4.1 Preparation: Bit-Fixing Predicates

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

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

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

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

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

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

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

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

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 $G_{\mathsf{auth}}^{\mathsf{BF}}(v_T) = \{00110, 10110\}$. Here, with an abuse of notation, we define the function $G_{\mathsf{auth}}^{\mathsf{BF}}$ for all input lengths.

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

4.2 Construction

Let $\ell = \ell(\kappa)$, and $n = n(\kappa)$ be integer-valued positive polynomials of the security parameter κ and Q be any constant positive integer smaller than ℓ. Let $\mathcal{C}^{\mathsf{BF}} := \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}} := \{\mathcal{C}^{\mathsf{BF}}_{\ell(\kappa)}\}_{\kappa\in\mathbb{N}}$ 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 ℓ and output length n.

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 $\mathsf{K}_{T,w} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ and $\widehat{\mathsf{K}}_{T,w} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [\ell]^Q$ and $w \in \{0,1\}^Q$. Then it outputs the master key as

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

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

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

It then computes

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

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 $y \in \{0,1\}^n$.

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

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

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 $x \in \{0,1\}^\ell$, it first parses $(\widetilde{\mathsf{K}}_{T,w})_{T\in[\ell]^Q, w\in\{0,1\}^Q} \leftarrow \mathsf{K}_v$. It then uses the PRF keys included in the constrained key and computes

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

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

4.3 Correctness

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

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

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

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

Then, by definition of function $G_{\mathsf{auth}}^{\mathsf{BF}}$, we have $x_T \in G_{\mathsf{auth}}^{\mathsf{BF}}(v_T)$ for any $T \in [\ell]^Q$ since we have $C_{v_T}^{\mathsf{BF}}(x_T) = 1$ if $C_v^{\mathsf{BF}}(x) = 1$. In particular, we have

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

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

4.4 Pseudorandomness on Constrained Points

Theorem 4.2. *If the underlying PRF Π_{PRF} is adaptive 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. We show the theorem by considering the following sequence of games between an adversary A against the pseudorandomness on constrained points security game and the challenger. In the following, for simplicity, we say an adversary A against the CPRF pseudorandomness game. Below, let E_i denote the probability that $b' = b$ holds in Game_i. Recall that 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 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 A as follows:

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

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

$$\bigwedge_{i=1}^{\ell} \Big((v_i^{(j)} \overset{?}{=} x_i^\dagger) \bigvee (v_i^{(j)} \overset{?}{=} *) \Big) = 0 \quad \text{for all} \quad C_{v^{(j)}}^{\mathsf{BF}} \in S_{\mathsf{con}}. \tag{1}$$

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 $T^\dagger \xleftarrow{\$} [\ell]^Q$. Let us set $T^\dagger = (t_1, \cdots, t_Q)$. The challenger further samples $b_{t_j}^\dagger \xleftarrow{\$} \{0,1\}$ for all $j \in [Q]$. Let $b_{T^\dagger}^\dagger := b_{t_1} b_{t_2} \cdots b_{t_Q} \in \{0,1\}^Q$. Then, the challenger checks whether the following equation holds with respect to the constrained key queries and the challenge query made by the adversary A at the end of the game:

- The challenger aborts if there exists $j \in [Q]$ such that

$$\Big((v_{t_j}^{(j)} \neq b_{t_j}^\dagger) \bigwedge (v_{t_j}^{(j)} \neq *) \Big) = 0 \tag{2}$$

is satisfied.

- The challenger aborts if $x^\dagger$ does not satisfy

$$(b^\dagger_{T^\dagger} \overset{?}{=} x^\dagger_{T^\dagger}) = \bigwedge_{j \in [Q]} (b^\dagger_{t_j} \overset{?}{=} x^\dagger_{t_j}) = 1. \tag{3}$$

- The challenger aborts if $(T^\dagger, b^\dagger_{T^\dagger})$ chosen by the challenger does not equal to the *first pair* (with respect to some pre-defined order over $[\ell]^Q \times \{0,1\}^Q$ 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 $[\ell]^Q \times \{0,1\}^Q$ since $Q = O(1)$.[8]

When the challenger aborts, it substitutes the guess $\widehat{b}$ outputted by 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 $(T^\dagger, b^\dagger_{T^\dagger}) \in [\ell]^Q \times \{0,1\}^Q$ that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). Therefore, the event abort occurs with probability $1 - 1/(2\ell)^Q$. Furthermore, it can be seen that abort occurs independently from the view of A. Therefore, we have

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

where we used the fact that $\widehat{b}$ is randomly chosen and thus equals to b 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 $T^\dagger$ and $b^\dagger_{T^\dagger}$ at the beginning of the game and aborts as soon as either A makes a constrained key query $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_\ell$ that does not satisfy Equation (2) or a challenge query for $x^\dagger$ that does not satisfy Equation (3). Furthermore, it aborts if $(T^\dagger, b^\dagger_{T^\dagger})$ 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 $b = 0$. 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 $x^\dagger \in \{0,1\}^\ell$ as the challenge query, the challenger first checks whether the condition in Equation (3) holds. If not it aborts. Otherwise, it samples $b \overset{\$}{\leftarrow} \{0,1\}$. In case $b = 0$, it computes $\mathsf{CPRF.Eval}(\mathsf{K}, x^\dagger)$ as

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

using the master key

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

[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.

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

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

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

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

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 $y^\dagger$ for $b = 0$ and $b = 1$ are exactly the same since A has not made an evaluation query on $x^\dagger$ and $\mathsf{K}_{T^\dagger, b^\dagger_{T^\dagger}}$ is not given through any of the constrained key query. Concretely, $\bar{y}^\dagger$ is distributed uniform random regardless of whether $b = 0$ or $b = 1$ and thus the value of b is information theoretically hidden to A. Therefore, we have

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

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

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

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

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

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

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

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

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

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 A that distinguishes Game_2 and Game_3 with non-negligible probability ϵ'. We then construct an adversary B that breaks the pseudorandomness of Π_{PRF} with the same probability. The adversary B proceeds as follows.

At the beginning of the game B samples a random tuple $T^\dagger = (t_1, \cdots, t_Q) \xleftarrow{\$} [\ell]^Q$ and $b^\dagger_{t_j} \xleftarrow{\$} \{0,1\}$ for all $j \in [Q]$ as in the Game_2-challenger. Let $b^\dagger_{T^\dagger} := b_{t_1} b_{t_2} \cdots b_{t_Q} \in \{0,1\}^Q$. Then, it further samples

$\mathsf{K}_{T,w} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [\ell]^Q$ and $w \in \{0,1\}^Q$ *except* for $(T^\dagger, b^\dagger_{T^\dagger})$. It then sets the (simulated) master key $\mathsf{K}^\dagger$ as

$$\mathsf{K}^\dagger = \left((\mathsf{K}_{T,w})_{T \in [\ell]^Q, w \in \{0,1\}^Q \setminus \{(T^\dagger, b^\dagger_{T^\dagger})\}}, (\widehat{\mathsf{K}}_{T,w})_{T \in [\ell]^Q, w \in \{0,1\}^Q} \right).$$

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

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

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

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

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

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

- When A submits $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_\ell$ as the j^{th} ($j \in [Q]$) constrained key query, 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 A:

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

  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, B can prepare all the PRF keys since the condition in Equation (2) guarantees us that we have $b^\dagger_{T^\dagger} \notin G^{\mathsf{BF}}_{\mathsf{auth}}(v^{(j)}_{T^\dagger})$, or equivalently, $C^{\mathsf{BF}}_{v^{(j)}_{T^\dagger}}(b^\dagger_{T^\dagger}) = 0$. Namely, $\widetilde{\mathsf{K}}_{T^\dagger, b^\dagger_{T^\dagger}} = \widehat{\mathsf{K}}_{T^\dagger, b^\dagger_{T^\dagger}}$ and so $\mathsf{K}_{T^\dagger, b^\dagger_{T^\dagger}}$ is not included in $\mathsf{K}_{v^{(j)}}$.

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

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

This completes the description of B. It is easy to check that in case $b = 0$, B receives $\bar{y}^\dagger \leftarrow \mathsf{PRF.Eval}(\mathsf{K}_{T^\dagger, b^\dagger_{T^\dagger}}, x^\dagger)$, hence B simulates Game_2 perfectly. Otherwise in case $b = 1$, B receives $\bar{y}^\dagger \xleftarrow{\$} \{0,1\}^n$, hence B simulates Game_3 perfectly. Therefore, we conclude that 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. $\square$

This completes the proof. $\square$

Theorem 4.5. *If the underlying PRF Π_{PRF} is adaptive pseudorandom, then our 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:

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

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

- $\left(\widetilde{\mathsf{K}}^{(0)}_{T,w}\right)_{T\in[\ell]^Q, w\in\{0,1\}^Q}$ 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.
- $\left(\widetilde{\mathsf{K}}^{(1)}_{T,w}\right)_{T\in[\ell]^Q, w\in\{0,1\}^Q}$ 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, A cannot compare outputs under the constrained key and the real master key since 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. $\square$

5 CPRFs for t-CNF from Standard PRFs

In this section, we provide a construction of an adaptive 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 parameter. Similarly to the result in Section 4, our result implies the existence of such CPRFs from one-way functions [GGM86, HILL99].

5.1 Preparation: t-CNF Predicates

We first define 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 5.1 (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 $C^{t\text{-}\mathsf{cnf}} : \{0,1\}^n \to \{0,1\}$ such that $t \le n$ is a set of clauses $C^{t\text{-}\mathsf{cnf}} := \{(J, C_J)\}_{J\in\mathcal{S}}$ 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:*

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

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 $\mathcal{C}^{t\text{-}\mathsf{cnf}}_n$ 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) $\mathcal{C}^{\mathsf{BF}}_n$ is included in $\mathcal{C}^{t\text{-}\mathsf{cnf}}_n$ for any $t \ge 1$. 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 $\mathcal{C}^{t\text{-}\mathsf{cnf}}$ 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 $G^{\text{t-cnf}}_{\text{auth}}$ 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 $G^{\text{t-cnf}}_{\text{auth}}$ takes as input a t-CNF predicate $C^{t\text{-cnf}} = \{(J, C_J)\}_{J\in S}$ 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 $G^{\text{t-cnf}}_{\text{auth}}$ is computed as

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

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

Remark 5.2. 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.

5.2 Construction

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

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 $\mathsf{K}_{\mathbb{J},\mathbb{w}} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ and $\widehat{\mathsf{K}}_{\mathbb{J},\mathbb{w}} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ 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 $\mathcal{S} := \{(a_1, \cdots, a_t) \in [\ell]^t \mid a_1 < \cdots < a_t\}$.

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

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

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 $y \in \{0,1\}^n$.

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

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

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

$\mathsf{CPRF.ConstrainEval}(\mathsf{K}_C, x) :\to y$: On input the constrained key K_C and an input $x \in \{0,1\}^\ell$, it first parses $(\widetilde{\mathsf{K}}_{\mathbb{J},\mathrm{w}})_{\mathbb{J}\in\mathcal{S}^Q,\mathrm{w}\in(\{0,1\}^t)^Q} \leftarrow \mathsf{K}_C$. 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 $y \in \{0,1\}^n$.

5.3 Correctness

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

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

Now, for any $x \in \{0,1\}^\ell$ 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 $G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}$, we have $x_{\mathbb{J}} \in G_{\mathsf{auth}}^{\mathsf{t\text{-}cnf}}(C^{t\text{-}\mathsf{cnf}}, \mathbb{J})$ 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.

5.4 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 5.3. *If the underlying PRF Π_{PRF} is adaptive pseudorandom, then our 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 $Q = O(1)$.*

Proof. We show the theorem by considering the following sequence of games between an adversary A against the pseudorandomness on constrained points security game and the challenger. In the following, for simplicity, we say an adversary A against the CPRF pseudorandomness game. Below, let E_i denote the probability that $b' = b$ holds in Game_i. Recall that 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 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 A as follows:

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

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

$$\bigwedge_{J \in \mathcal{S}} C_J^{(k)}(x_J^\dagger) = 0 \quad \text{for all} \quad C^{t\text{-}\mathsf{cnf}^{(k)}} \in S_{\mathsf{con}}, \tag{7}$$

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 $\mathbb{J}^\dagger = (J_1, \cdots, J_Q) \xleftarrow{\$} \mathcal{S}^Q$. The challenger further samples $\mathbb{w}^\dagger = (w_1, \cdots, w_Q) \xleftarrow{\$} (\{0,1\}^t)^Q$ 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 A at the end of the game:

- The challenger aborts if there exists $k \in [Q]$ such that

$$C^{t\text{-}\mathsf{cnf}^{(k)}}_{J_k}(w_k) = 1 \tag{8}$$

  is satisfied.
- The challenger aborts if $x^\dagger$ does not satisfy

$$(\mathbb{w}^\dagger \overset{?}{=} x^\dagger_{\mathbb{J}^\dagger}) = \bigwedge_{k \in [Q]} (w_k \overset{?}{=} x^\dagger_{J_k}) = 1 \tag{9}$$

- The challenger aborts if $(\mathbb{J}^\dagger, \mathbb{w}^\dagger)$ 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 (8) for all $k \in [Q]$ and Equation (9). Note that it is possible to efficiently find such a pair by enumerating over $\mathcal{S}^Q \times (\{0,1\}^t)^Q$ since $t, Q = O(1)$.[9]

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

As we will show in Lemma 5.4, there exists at least a single pair $(\mathbb{J}^\dagger, \mathbb{w}^\dagger) \in \mathcal{S}^Q \times (\{0,1\}^t)^Q$ that satisfies Equation (8) for all $k \in [Q]$ and Equation (9). Therefore, the event abort occurs with probability $1 - 1/(|\mathcal{S}| \cdot 2^t)^Q$ where $|\mathcal{S}| = \binom{\ell}{t}$. Furthermore, it can be seen that abort occurs independently from the view of 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 $\widehat{b}$ is randomly chosen and thus equals to b with probability $1/2$ when abort occurs. As in Remark 5.2, 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 $\mathbb{J}^\dagger$ and $\mathbb{w}^\dagger$ at the beginning of the game and aborts as soon as either A makes a constrained key query $C^{t\text{-}\mathsf{cnf}^{(k)}} \in \mathcal{C}_\ell^{t\text{-}\mathsf{cnf}}$ that does not satisfy Equation (8) or a challenge query for $x^\dagger$ that does not satisfy Equation (9). Furthermore, it aborts if $(\mathbb{J}^\dagger, \mathbb{w}^\dagger)$ is not the first pair that satisfies Equation (8) for all $k \in [Q]$ and Equation (9). Since this is only a conceptual change, we have

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

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

Game_3**:** In this game, we change how the challenger responds to the challenge query when $b = 0$. 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 $x^\dagger \in \{0,1\}^\ell$ as the challenge query, the challenger first checks whether the condition in Equation (9) holds. If not, it aborts. Otherwise, it samples $b \xleftarrow{\$} \{0,1\}$. In case $b = 0$, it computes $\mathsf{CPRF.Eval}(\mathsf{K}, x^\dagger)$ as

$$y = \bigoplus_{\mathbb{J} \in \mathcal{S}^Q} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}, x_{\mathbb{J}}}, x), \tag{10}$$

using the master key

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

that it constructed at the beginning of the game, where $\mathsf{K}_{\mathbb{J},\mathrm{w}}, \widehat{\mathsf{K}}_{\mathbb{J},\mathrm{w}} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ for all $\mathbb{J} \in \mathcal{S}^Q$ and $w \in (\{0,1\}^t)^Q$. Due to the condition in Equation (9), i.e., $\bigwedge_{k \in [Q]} (w_k \overset{?}{=} x^\dagger_{J_k}) = 1$, we can rewrite Equation (10) as

$$y^\dagger = \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}^\dagger, \mathrm{w}^\dagger}, x^\dagger) \oplus \left(\bigoplus_{\mathbb{J} \in \mathcal{S}^Q \setminus \{\mathbb{J}^\dagger\}} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}, x^\dagger_{\mathbb{J}}}, x^\dagger) \right). \tag{11}$$

In this game Game_3, when $b = 1$, the challenger instead samples a random $\bar{y}^\dagger \xleftarrow{\$} \{0,1\}^n$ and returns the following to A instead of returning $y^\dagger$ to A as in Equation (11):

$$y^\dagger = \bar{y}^\dagger \oplus \left(\bigoplus_{\mathbb{J} \in \mathcal{S}^Q \setminus \{\mathbb{J}^\dagger\}} \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}, x^\dagger_{\mathbb{J}}}, x^\dagger) \right). \tag{12}$$

We show in Lemma 5.5 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 $y^\dagger$ for $b = 0$ and $b = 1$ are exactly the same since A has not made an evaluation query on $x^\dagger$ and $\mathsf{K}_{\mathbb{J}^\dagger, \mathrm{w}^\dagger}$ is not given through any of the constrained key query. Concretely, $\bar{y}^\dagger$ is distributed uniform random regardless of whether $b = 0$ or $b = 1$ and thus the value of b is information theoretically hidden to A. Therefore, we have

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

Combining everything together with Lemma 5.4 and Lemma 5.5, we have

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

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

Lemma 5.4. *In* Game_1*, we have*

$$\left\{ (\mathbb{J}^\dagger, \mathrm{w}^\dagger) \in \mathcal{S}^Q \times (\{0,1\}^t)^Q \;\middle|\; \begin{array}{l} (\mathbb{J}^\dagger, \mathrm{w}^\dagger) \textit{ satisfies Equation } (8) \\ \textit{for all } k \in [Q], \textit{ and Equation } (9) \end{array} \right\} \neq \emptyset.$$

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

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

Let us denote $\bar{\mathbb{J}} := (J^{(1)}, \cdots, J^{(Q)}) \in \mathcal{S}^Q$ and $\bar{\mathbb{w}} := (x^{\dagger}_{J^{(1)}}, \cdots, x^{\dagger}_{J^{(Q)}}) \in (\{0,1\}^t)^Q$. It is easy to check that Equation (8) for all $k \in [Q]$ and Equation (9) hold if $\mathbb{J}^{\dagger} = \bar{\mathbb{J}}$ and $\mathbb{w}^{\dagger} = \bar{\mathbb{w}}$. □

Lemma 5.5. *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 A that distinguishes Game_2 and Game_3 with non-negligible probability ϵ'. We then construct an adversary B that breaks the pseudorandomness of Π_{PRF} with the same probability. The adversary B proceeds as follows.

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

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

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

- When A submits $x \in \{0,1\}^{\ell}$ as the evaluation query, B checks whether $x_{\mathbb{J}^{\dagger}} = \mathbb{w}^{\dagger}$. If not, then it can use the simulated master key $\mathsf{K}^{\dagger}$ 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, B computes the output as

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

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

- When A submits $C^{t\text{-}\mathsf{cnf}(k)} \in \mathcal{C}^{t\text{-}\mathsf{cnf}}_{\ell}$ as the k^{th} ($k \in [Q]$) constrained key query, B checks whether the condition in Equation (8) holds. If not it aborts and outputs a random bit. Otherwise, it returns the following constrained key $\mathsf{K}_{C^{(k)}}$ to 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 $\mathbb{w} \in G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}(k)}, \mathbb{J}) = \{\mathbb{w} \in (\{0,1\}^t)^Q \mid \bigwedge_{i\in[Q]} C^{(k)}_{J_i}(w_i) = 1\}$ and $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{w}} = \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{w}}$ otherwise. Here, B can prepare all the PRF keys since the condition in Equation (8) guarantees us that we have $\mathbb{w}^{\dagger} \notin G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}(k)}, \mathbb{J})$, or equivalently, $\bigwedge_{i\in[Q]} C^{(k)}_{J_i}(w_i) = 0$. Namely, $\widetilde{\mathsf{K}}_{\mathbb{J}^{\dagger},\mathbb{w}^{\dagger}} = \widehat{\mathsf{K}}_{\mathbb{J}^{\dagger},\mathbb{w}^{\dagger}}$ and so $\mathsf{K}_{\mathbb{J}^{\dagger},\mathbb{w}^{\dagger}}$ is not included in $\mathsf{K}_{C^{(k)}}$.

- When A submits the target input $x^\dagger \in \{0,1\}^\ell$ as the challenge query, B checks whether the condition in Equation (9) holds. If not it aborts and outputs a random bit. Otherwise, B queries its PRF challenger on $x^\dagger$ as its challenge query and receives back $\bar{y}^\dagger$. It then computes $y^\dagger$ as in Equation (12) and returns $y^\dagger$ to A. Here, since Equation (9) holds, $\mathsf{K}_{\mathbb{J}^\dagger,\mathbb{w}^\dagger}$ must be required to compute on input $x^\dagger$.

Finally, A outputs its guess b'. B then checks whether $(\mathbb{J}^\dagger, \mathbb{w}^\dagger)$ is the first pair that satisfies Equation (8) for all $k \in [Q]$ and Equation (9). If it does not hold, B outputs a random bit. Otherwise, B outputs b' as its guess.

This completes the description of B. It is easy to check that in case $b = 0$, B receives $\bar{y}^\dagger \leftarrow \mathsf{PRF.Eval}(\mathsf{K}_{\mathbb{J}^\dagger,\mathbb{w}^\dagger}, x^\dagger)$, hence B simulates Game_2 perfectly. Otherwise in case $b = 1$, B receives $\bar{y}^\dagger \xleftarrow{\$} \{0,1\}^n$, hence B simulates Game_3 perfectly. Therefore, we conclude that 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. $\square$

This completes the proof. $\square$

Theorem 5.6. *If the underlying PRF Π_{PRF} is adaptive 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:

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

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 A queries. Then, A receives either one of the following two distributions:

- $\Big(\widetilde{\mathsf{K}}^{(0)}_{\mathbb{J},\mathbb{w}} \Big)_{\mathbb{J} \in \mathcal{S}^Q, \mathbb{w} \in (\{0,1\}^t)^Q}$ where $\widetilde{\mathsf{K}}^{(0)}_{\mathbb{J},\mathbb{w}} = \mathsf{K}_{\mathbb{J},\mathbb{w}}$ if and only if $\mathbb{w} \in G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}^{(0)}}, \mathbb{J})$, and $\widetilde{\mathsf{K}}_{\mathbb{J},\mathbb{w}} = \widehat{\mathsf{K}}_{\mathbb{J},\mathbb{w}}$ otherwise.
- $\Big(\widetilde{\mathsf{K}}^{(1)}_{\mathbb{J},\mathbb{w}} \Big)_{\mathbb{J} \in \mathcal{S}^Q, \mathbb{w} \in (\{0,1\}^t)^Q}$ where $\widetilde{\mathsf{K}}^{(1)}_{\mathbb{J},\mathbb{w}} = \mathsf{K}_{\mathbb{J},\mathbb{w}}$ if and only if $\mathbb{w} \in G^{\mathsf{t\text{-}cnf}}_{\mathsf{auth}}(C^{t\text{-}\mathsf{cnf}^{(1)}}, \mathbb{J})$, 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, A cannot compare outputs under the constrained key and the real master key since 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. $\square$

6 Conclusion and future work

In conclusion, we have developed the first adaptively-secure CPRF construction with $O(1)$ collusion-resistance for t-CNF predicates, which in particular includes as a special case the bit-fixing predicates. Our technique signals a noted departure from existing techniques and uses much weaker assumption than any previous constructions for comparable predicates. Our construction is the first to achieve adaptive security and/or collusion-resistance in the standard model and without strong assumptions (e.g., IO, multilinear maps) for non-trivial predicates. Finally our construction achieves 1-key privacy for free.

We believe that there are a number of interesting future directions for our work. Since our technique is substantially different from existing constructions, we believe that our methods could widen the possibilities for constructing CPRFs. We summarise the most interesting avenues in the following items.

- Adapt our technique to construct adaptively-secure constant collusion-resistant CPRFs for bounded-depth circuits. Here, even constructing CPRFs for inner products already seems to require new ideas as it requires some sort of algebraic structure absent in standard PRFs.
- Devise a construction that satisfies $p(\kappa)$-collusion-resistance for some possibly bounded polynomial $p(\kappa)$ for bit-fixing predicates.
- Satisfy key privacy for > 1 constrained keys for bit-fixing predicates.

The first point would immediately give a more expressive CPRF. The second point would lead to applications (such as those envisioned by [BW13]) with far more utility. Since the number of constrained keys would be linked to the size of the security parameter. As for the last point, we remark that constructing CPRF for r-key privacy for $r > 1$ supporting *general circuits* from a standard assumption seems to be out of our current reach since it immediately implies IO for general circuits via the results of Canetti and Chen [CC17]. However, CPRF for r-key privacy for *bit-fixing predicates* may be much easier to construct. It merely implies IO for bit-fixing predicates, which is trivial to construct since the bit-fixing predicates have an efficiently computable standard form. We also remark that distributional VBB obfuscation for bit-fixing predicates is a different thing from IO for bit-fixing predicates in that its security is defined in a situation where the predicate is chosen randomly following certain distribution. As for the distributional VBB obfuscation, we know several constructions from various assumptions [BVWW16, WZ17, GKW17, BLMZ18] and even unconditionally [BLMZ18] in certain regime of parameters.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572. (Cited on page 4.)

[AMN$^+$18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Shacham and Boldyreva [SB18], pages 543–574. (Cited on page 3, 4, 5.)

[AMN$^+$19] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively single-key secure constrained PRFs for NC^1. In *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*, pages 223–253, 2019. (Cited on page 5.)

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Dodis and Nielsen [DN15], pages 31–60. (Cited on page 2, 5, 11, 12.)

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401, 2013. http://eprint.iacr.org/2013/401. (Cited on page 3, 4.)

[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, 3, 5.)

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Kalai and Reyzin [KR17], pages 567–594. (Cited on page 5.)

[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 6.)

[BLMZ18] James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. New techniques for obfuscating conjunctions. *IACR Cryptology ePrint Archive*, 2018:936, 2018. (Cited on page 25.)

[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 2, 3, 4, 5, 9, 11, 12.)

[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, 4, 5.)

[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 Dodis and Nielsen [DN15], pages 1–30. (Cited on page 2, 3, 5.)

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In Madhu Sudan, editor, *ITCS 2016*, pages 147–156. ACM, January 2016. (Cited on page 25.)

[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, 5, 12, 25.)

[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, 4, 5, 9, 11, 25.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552. (Cited on page 4.)

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Shacham and Boldyreva [SB18], pages 577–607. (Cited on page 2, 3, 4, 5.)

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, March 2015. (Cited on page 25, 26.)

[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.)

[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 3, 4.)

[FOC17] *58th FOCS*. IEEE Computer Society Press, 2017. (Cited on page 27, 28.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. (Cited on page 2, 3, 7, 10, 12, 18.)

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Kalai and Reyzin [KR17], pages 537–566. (Cited on page 5.)

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May / June 2010. (Cited on page 25, 26.)

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In FOCS 2017 [FOC17], pages 612–621. (Cited on page 25.)

[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 7, 10, 12, 18.)

[HKKW14] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. Cryptology ePrint Archive, Report 2014/720, 2014. http://eprint.iacr.org/2014/720. (Cited on page 2, 5.)

[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 5.)

[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 3, 4.)

[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 13*, pages 669–684. ACM Press, November 2013. (Cited on page 2, 3, 4, 5.)

[KR17] Yael Kalai and Leonid Reyzin, editors. *TCC 2017, Part II*, volume 10678 of *LNCS*. Springer, Heidelberg, November 2017. (Cited on page 25, 27.)

[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 *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*, pages 158–188, 2019. (Cited on page 4.)

[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.)

[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, 5.)

[SB18] Hovav Shacham and Alexandra Boldyreva, editors. *CRYPTO 2018, Part II*, volume 10992 of *LNCS*. Springer, Heidelberg, August 2018. (Cited on page 25, 26.)

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. Cryptology ePrint Archive, Report 2019/365, 2019. https://eprint.iacr.org/2019/365. (Cited on page 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 9, 15.)

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In FOCS 2017 [FOC17], pages 600–611. (Cited on page 25.)