Constrained PRFs for Bit-fixing from OWFs with Constant Collusion Resistance*

Alex Davidson[1], Shuichi Katsumata[2,4], Ryo Nishimaki[3], Shota Yamada[4]

[1]ISG, Royal Holloway University of London, UK
alex.davidson.2014@rhul.ac.uk
[2]The University of Tokyo, Tokyo, Japan
shuichi_katsumata@it.k.u-tokyo.ac.jp
[3]NTT Secure Platform Laboratories, Tokyo, Japan
nishimaki.ryo@lab.ntt.co.jp
[4]National Institute of Advanced Industrial Science and Technology (AIST)
yamada-shota@aist.go.jp

October 31, 2018

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, a requirement for all 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 predicates. CPRFs from more standard assumptions only satisfy security for a single constrained key query.

In this work, we give the first construction of a CPRF that can issue a constant number of constrained keys for bit-fixing 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), and that it also achieves fully adaptive security. This is the only construction to achieve adaptive security outside of the random oracle model, and without sub-exponential security losses. Our technique represents a noted departure from existing CPRF constructions. We hope that it may lead to future constructions that can expose a greater number of keys, or consider more expressive predicates (such as bounded-depth circuit constraints).

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

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

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

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. That is, let $\mathcal{X}$ denote the input space, and let $S \subseteq \mathcal{X}$. Then a constrained key K_S for CPRF allows evaluating $\mathsf{CPRF.Eval}(\mathsf{K}, x)$ if and only if $x \in S$. In the security game, the adversary is permitted to make queries for learning PRF evaluations as with standard PRFs. The adversary is also permitted to learn constrained keys for any subsets $S \subseteq \mathcal{X}$ that it wants. Security now dictates that the CPRF remains pseudorandom on an input point that lies outside of the queried subsets. If an adversary can ask more than one constrained key query, then we say the CPRF is *collusion-resistant*.

In this work, our main question is:

Can we construct constrained PRFs with collusion-resistance from standard assumptions?

Up until now, this question has been unanswered in either the affirmative or the negative. An extra consideration that we have to make is what kind of predicates that we consider for our CPRF to satisfy.

PREDICATES. While constrained keys can be defined with respect to subsets, a more natural definition defines functionality with respect to predicates. That is, the constrained key allows evaluation of the function on x, if and only if the associated predicate is equal to 1 on x. We denote such a predicate by P and $1 \leftarrow P(x)$ indicates that the input satisfies the constraint. Otherwise, we write $0 \leftarrow P(x)$.

Many suitable predicates for CPRFs have been proposed in the literature, such as:

- puncturing [BW13, KPTZ13, BGI14, BLW17];
- prefixes [BW13, BFP+15];
- left-right (LR) [BW13];
- bit-fixing (BF) [BW13, BLW17, CC17, AMN+18];
- general circuits in NC^1 [CC17, AMN+18, CVW18];
- general circuits in $\mathsf{P/poly}$ [BW13, HKKW14, BV15, BLW17, BTVW17, PS18].

We highlight the two predicates that are the most interesting due to the expressibility and flexibility of the predicate class that they support.

- Bit-fixing (BF) predicates are associated with a string $v \in \{0, 1, *\}^\ell$ as input; where $v_i = *$ indicates a *wildcard* entry. Denote the predicate by $P_v(x)$ for some $x \in \{0, 1\}^\ell$. Then we say that $1 \leftarrow P_v(x)$ iff $(x_i = v_i) \vee (v_i = *)$ for each $i \in [\ell]$.
- General circuit predicates are associated with some representative circuit $C \in \mathcal{C}$. We say that $1 \leftarrow P_C(x)$ for $x \in \{0, 1\}^\ell$ if and only if $C(x) = 1$.

m-KEY PRIVACY. 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 negligible advantage. A CPRF satisfying this definition of security is known as a private CPRF or PCPRF.[1] The definition can be made stronger by requiring that the adversary is given m keys for $m \geq 1$, and also by allowing the adversary access to the evaluation oracle. When no evaluation queries are permitted, the security guarantee is referred to as *weak* key privacy. The applications that [BLW17] consider are all satisfied by the weaker notion of privacy; so proving security with respect to this definition is enough. As a consequence, when we refer to key privacy, we will always refer to the weaker definition.

It was shown by Canetti and Chen [CC17] that a CPRF satisfying m-key privacy for $m \geq 2$ and bounded-depth circuit predicates implies the existence of IO. This implication holds using the weaker notion of privacy, also. In [CC17], they also show that security in the simulation-based setting is stronger than in the indistinguishability model of [BLW17]. In the simulation-based security framework, a CPRF satisfying m-key privacy implies the existence of VBB obfuscation. Therefore, it is impossible to construct CPRFs for P/poly with m-key privacy (for $m \geq 2$) in this setting [BGI+12].

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. It was observed in the original works that the GGM-PRF [GGM86] can be used as a CPRF for the very simply puncturing or prefix-fixing predicates. Put differently, construction of CPRFs for such simple predicates are known to exist from OWFs. On the other hand, CPRFs supporting more flexible predicates such as LR, bit-fixing, and 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 ROM or strong assumptions, these CPRFs for flexible predicates satisfy collusion-resistance for any polynomial number of constrained keys.

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) general circuit constraints either for NC^1 or P/poly. However, all of these constructions do not satisfy collusion resistance. The work of Attrapadung et al. [AMN+18] provides CPRFs for BF and NC^1 from traditional groups. However, their constructions too do not satisfy collusion resistance. Moreover, it is worthwhile to mention that all the CPRFs listed above only achieve selective security in the standard model.

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

1.1.1 Achieving private constraints.

The constructions of [BLW17] satisfy poly-key privacy for circuit predicates under the existence of IO. The PCPRFs of [CC17, BTVW17, PS18, CVW18] also satisfy the privacy guarantee for circuit predicates, but only in the case of $m = 1$. Achieving privacy for $m > 1$ seems challenging, since it would imply the existence of IO for P/poly from LWE [CC17]. Finally, CPRF for LR 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] They are also known as 'constraint-hiding' CPRFs.

1.2 Our contribution

In this work, we develop a new CPRF construction for the bit-fixing predicate. While this predicate may be 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 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 (i.e. a constant number with respect to the security parameter). Additionally, our security proof holds in the setting where *all* queries are made adaptively, with only a polynomial loss in security. Some previous constructions satisfy adaptive security from non-standard assumptions or within the ROM. Otherwise, they can only satisfy adaptive security under sub-exponential security losses.

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 $m > 1$. We leave this open as a future research direction, since such a construction would imply obfuscation for bit-fixing predicates from OWFs. We summarize our contribution alongside the previous state-of-the-art in Table 1.

Table 1: List of existing constructions of CPRFs along with their functionality and the assumptions required. The adaptive column only refers to works that achieve adaptive security in polynomial-time. We categorize [KPTZ13, BGI14] as puncturing CPRFs since they use GGM-based techniques. We do not consider the CPRFs of [Bit17, GHKW17] since they do not permit evaluation queries.

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

1.3 Roadmap

In Section 2 we give a technical overview of our scheme. In Section 3, we discuss the preliminaries required for consuming our construction. In Section 4 we give our CPRF construction along with the proofs of correctness and security. We conclude this work in Section 5.

2 Technical overview

2.1 Lattice-based constructions

The idea for this work originates from the lattice-based CPRF for bit-fixing constraints of Canetti and Chen [CC17]. In these works the adversary is allowed to query for one constrained key that is chosen selectively (rather than adaptively). The PRF is defined over an input $x \in \{0,1\}^\ell$ and the master secret key is a set of Gaussian-distributed matrices $\{\boldsymbol{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$. These matrices are thought of as representatives of LWE secrets, the underlying technique is borrowed from the PRFs of [BPR12, BLMR13]. The constrained key for some bit-fixing predicate $v \in \{0,1,*\}^\ell$ is a set of matrices, where we reveal $\boldsymbol{D}_{i,v_i}$ if $v_i \in \{0,1\}$, and reveal both $\{\boldsymbol{D}_i\}_{b\in b\in\{0,1\}}$ if $v_i = *$, for each $i \in [\ell]$. Finally, newly sampled $\overline{\boldsymbol{D}_{i,1-v_i}} \xleftarrow{\$} \chi^{m\times m}$ replace the matrices that are not learnt.[2] In the public parameters, there is a matrix $\boldsymbol{A}$, and for an input x, the PRF evaluation is the product $\boldsymbol{A} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,x_i}$, rounded to some appropriate integer $p > 0$.

The key observation of [CC17] is that pseudorandomness only has to hold for some challenge $x^\dagger$ where $(x^\dagger_j \neq v_j) \wedge (v_j \neq *)$ and v is the selectively chosen bit-fixing predicate at the beginning of the security game. Then when the PRF is evaluated at $x^\dagger$, the output includes the matrix $\boldsymbol{D}_{j,x^\dagger_j}$ in the product. This matrix is not revealed in the constrained key, and thus not revealed to the adversary either. As a result, their security proof follows the formula of [BLMR13]; relying on an LWE security reduction where $\boldsymbol{D}_{j,x^\dagger_j}$ ultimately acts as an unknown LWE secret. It is also noted by [CC17] that a very similar argument can be used to show that the [BLMR13] PRF is also a PCPRF for bit-fixing constraints. For circuit-based constraints, this proof technique does not apply since the matrices are no longer tied explicitly to one bit of the constraint query. In these cases, a more careful LWE argument is used with the secret distribution that is considered.

Unfortunately, the analysis for bit-fixing does not follow for more than one key. It is entirely possible to choose two constrained keys that would reveal the entire set $\{\boldsymbol{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$, without compromising all evaluation points.[3] Therefore, the LWE argument cannot be used since all the secrets are effectively revealed to the adversary. The main issue of this technique is that one bit of the PRF input is tied concretely to one matrix in the master secret key. Consequently, when valid constraints reveal both components of the master secret key for each bit, security is effectively lost.

2.2 Our scheme

To improve on the functionality of previous schemes, we design a CPRF construction that analyses Q input bits at a time, for $Q \geq 1$. That is to say that [CC17] analyses only 1 input bit at a time, and we regard this as an important distinction. We also depart from the lattice constructions above, effectively replacing the matrices above with keys to an underlying PRF. It becomes clear that we require $Q = O(1)$ later, as the size of the master secret key is $O(\mathsf{poly}(\kappa)^Q)$.

KEY GENERATION. To be more precise, let PRF be an underlying pseudorandom function, where $\mathsf{PRF.Eval} : \{0,1\}^\kappa \times \{0,1\}^{n_{\mathsf{in}}} \mapsto \{0,1\}^{n_{\mathsf{out}}}$. Let $T = (t_1, \ldots, t_Q)$ denote a tuple of Q values taken

[2] This is not required for standard CPRF security, but only for the extra privacy property.

[3] For example, choosing the constraints $v_1 = 1***1$ and $v_2 = 0***0$; where $x^\dagger = 1***0$ is still a constrained point.

from $[n_{\mathsf{in}}]$, that is $T \in [n_{\mathsf{in}}]^Q$.

The 'functional' master secret key is generated by running the following loop.

- For $T \in [n_{\mathsf{in}}]^Q$ and $w \in \{0,1\}^Q$: sample $\overline{\mathsf{K}}_{T,w} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$.

Clearly, we require that $Q = O(1)$, otherwise this loop would run in super-polynomial time with respect to κ. Finally, let $\overline{\mathsf{K}} = \{\overline{\mathsf{K}}_{T,w}\}_{T\in[n_{\mathsf{in}}]^Q, w\in\{0,1\}^Q}$.

We also run a separate invocation of the above to generate a 'dummy' set of PRF keys. That is, for $T \in [n_{\mathsf{in}}]^Q$ and $w \in \{0,1\}^Q$: sample $\widehat{\mathsf{K}}_{T,w} \xleftarrow{\$} \mathsf{PRF.Gen}(1^\kappa)$. Let $\widehat{\mathsf{K}} = \{\widehat{\mathsf{K}}_{T,w}\}_{T\in[n_{\mathsf{in}}]^Q, w\in\{0,1\}^Q}$.

The entire master secret key is then given by $\mathsf{K} = (\overline{\mathsf{K}}, \widehat{\mathsf{K}})$. The need for the dummy key becomes apparent soon, but essentially this is to allow us to satisfy 1-key privacy.

Evaluation. To evaluate the CPRF on $x \in \{0,1\}^{n_{\mathsf{in}}}$, do the following.

- For each $T \in [n_{\mathsf{in}}]^Q$, let x_T be the string $x_{t_1}x_{t_2}\cdots x_{t_Q}$.
- Compute the XOR: $y \leftarrow \bigoplus_{T\in[n_{\mathsf{in}}]^Q} \mathsf{PRF.Eval}(\overline{\mathsf{K}}_{T,x_T}, x)$.
- Output y.

In other words, evaluation requires $(n_{\mathsf{in}})^Q$ evaluations of the underlying PRF. Recall that $Q = O(1)$ and so it also runs in polynomial time.

Constraining. To produce constrained keys for the function, we only analyse bit-fixing predicates implied by strings $v \in \{0,1,*\}^{n_{\mathsf{in}}}$. Here, we use the notation that $*$ is a wildcard character, and we say that a binary string $x \in \{0,1\}^{n_{\mathsf{in}}}$ satisfies the bit-fixing predicate C_v^{BF}, with respect to v, if and only if:

$$\bigwedge_{i=1}^{n_{\mathsf{in}}} \Big((v_i \stackrel{?}{=} x_i) \bigvee (v_i \stackrel{?}{=} *) \Big) = 1.$$

We may also write $b \leftarrow C_v^{\mathsf{BF}}(x)$, where $b = 1$ indicates satisfaction and $b = 0$ indicates otherwise.

Now let $v_T = v_{t_1}v_{t_2}\cdots v_{t_Q}$ be defined as before, for $T = (t_1, \ldots, t_Q) \in [n_{\mathsf{in}}]^Q$. Then to produce a constrained key, K_v for our CPRF we analyse each string v_T individually against all possible $w \in \{0,1\}^Q$.

Recall that the definition of a CPRF states that evaluation should remain the same using K_v, if $C_v^{\mathsf{BF}}(x) = 1$. Before we describe constraining for general parameter settings, it may be helpful to consider a small example where $n_{\mathsf{in}} = n_{\mathsf{out}} = 8$, $Q = 3$, $v = 100**1*0$ and let $T = (3,4,6)$. Then $v_T = 0*1 \in \{0,1,*\}^Q$, and so we should be able to evaluate the CPRF on inputs, where $x_T = 001$ or $x_T = 011$. To achieve this, we can let $\widetilde{\mathsf{K}}^v_{T,001} = \overline{\mathsf{K}}_{T,001}$ and $\widetilde{\mathsf{K}}^v_{T,011} = \overline{\mathsf{K}}_{T,011}$. However, we do not have to reveal any keys where $C_{v_T}^{\mathsf{BF}}(x_T) = 0$; e.g. if $x_T = 100$. In these occurrences, we let $\widetilde{\mathsf{K}}^v_{T,w} = \widehat{\mathsf{K}}_{T,w}$ (that is, dummy keys). Finally, we let:

$$\mathsf{K}_v = \Big(\{\widetilde{\mathsf{K}}^v_{T,001}\} \cup \{\widetilde{\mathsf{K}}^v_{T,011}\} \cup \{\widetilde{\mathsf{K}}^v_{T,w}\}_{w\notin\{001,011\}} \Big)_{T\in[n_{\mathsf{in}}]^Q},$$

denote the entire constrained key.

In general, we define the constraining algorithm to do the following.

- For all $T = (t_1, \ldots, t_Q) \in [n_{\mathsf{in}}]^Q$: let $v_T = v_{t_1}v_{t_2}\cdots v_{t_Q}$.
- For $w \in \{0,1\}^Q$, if $C_{v_T}^{\mathsf{BF}}(w) = 1$: let $\widetilde{\mathsf{K}}^v_{T,w} = \overline{\mathsf{K}}_{T,w}$.
- For $w \in \{0,1\}^Q$, if $C_{v_T}^{\mathsf{BF}}(w) = 0$: let $\widetilde{\mathsf{K}}^v_{T,w} = \widehat{\mathsf{K}}_{T,w}$.
- Output $\mathsf{K}_v = \Big(\widetilde{\mathsf{K}}^v_{T,w}\Big)_{T\in[n_{\mathsf{in}}]^Q, w\in\{0,1\}^Q}$.

CONSTRAINED EVALUATION. An interesting property of our scheme is that constrained keys are essentially distributed the same as master secret keys where the dummy portion of the key is removed (i.e., $\mathsf{K} = (\overline{\mathsf{K}}, \emptyset)$). This is because all the underlying set elements are just keys sampled from the keyspace of PRF. Therefore, constrained evaluation is identical to the real evaluation algorithm.

2.3 Proving security

CONSTRAINED PSEUDORANDOMNESS. The main goal of our work is to prove that our construction satisfies pseudorandomness on "constrained points" after receiving $Q = O(1)$ constrained keys. Let $v^{(1)}, \ldots, v^{(Q)}$ be the bit-fixing strings that the constrained keys are defined with respect to.

Our goal is to essentially show that there exists a $T^\dagger = (t_1^\dagger, \ldots, t_Q^\dagger)$ such that $v^{(i)}_{t_i^\dagger} \neq *$ for $i \in [Q]$. If this was not the case, then this implies that there exists $i \in [Q]$ such that $v^{(i)} = * * \cdots *$ (i.e. the all wildcard string). If this was true, then any eventual challenge input point $x^\dagger$ would be unconstrained (since $1 \leftarrow C^{\mathsf{BF}}_{v^{(i)}}(x^\dagger)$), and thus the security game is void.

For each $v^{(i)}$, let $\mathsf{K}_{v^{(i)}} = \left(\widetilde{\mathsf{K}}^{(i)}_{T,w}\right)_{T \in [n_{\mathsf{in}}]^Q, w \in \{0,1\}^Q}$. Now consider the string:

$$w^\dagger = (1 - v^{(1)}_{t_1^\dagger})\,(1 - v^{(2)}_{t_2^\dagger}) \cdots (1 - v^{(Q)}_{t_Q^\dagger}).$$

Then $C^{\mathsf{BF}}_{v^{(i)}_{T^\dagger}}(w^\dagger) = 0$ for all $i \in [Q]$, and thus $\widetilde{\mathsf{K}}^{(i)}_{T^\dagger, w^\dagger} = \widehat{\mathsf{K}}_{T^\dagger, w^\dagger}$. Therefore, the 'functional' key $\overline{\mathsf{K}}_{T^\dagger, w^\dagger} \in \mathsf{K}$ is never revealed by a constrained key query. This is the heart of our security proof.

Our security reduction uses the fact that constrained evaluations can be handled by a reduction algorithm that has access to an oracle for the underlying PRF at a randomly chosen key $\mathsf{K}^\dagger$. Namely, the reduction algorithm simulates the master secret key in its entirety, apart from for the key $\overline{\mathsf{K}}_{T^\dagger, w^\dagger}$ and simulates the challenger for the CPRF adversary. Consider an evaluation query x made by the CPRF adversary that needs to use the key $\overline{\mathsf{K}}_{T^\dagger, w^\dagger}$, this implies that $x_T = w^\dagger$. Then the query is answered by submitting x as an oracle query in the PRF game, where $\overline{\mathsf{K}}_{T^\dagger, w^\dagger} = \mathsf{K}^\dagger$ is implicitly chosen. For the challenge constrained input $x^\dagger$ output by the CRPF adversary, the reduction algorithm submits $x^\dagger$ to its challenge oracle and receives back $y^\dagger \leftarrow \mathsf{PRF.Eval}(\mathsf{K}^\dagger, x^\dagger)$ or $y^\dagger \leftarrow f(x^\dagger)$ for a uniformly sampled function $f : \{0,1\}^{n_{\mathsf{in}}} \mapsto \{0,1\}^{n_{\mathsf{out}}}$.[4]

The fact that the value $y \in \{0,1\}^{n_{\mathsf{out}}}$ output by the CPRF is pseudorandom is obtained via the fact that $y^\dagger$ is XORed into y, and $y^\dagger$ is a pseudorandom output dependent solely on $x^\dagger$ (by the security of PRF). See Theorem 4.1 for the full proof of security.

ADAPTIVE SECURITY. Our construction arrives at adaptive security *for free*. Previous constructions incur sub-exponential security losses during the reduction from adaptive to selective security. Essentially all constructions use the technique where the adaptive adversary attempts to guess the challenge point $x^\dagger$ that the selective 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. $K_{T^\dagger, w^\dagger}$). 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 4.2. Similar problems were identified by Waters [Wat05] who introduced the "artificial abort step".

[4]Note that we require a version of PRF security where challenge points and standard evaluation queries are distinct in the sense that evaluation queries are always answered by the actual PRF.

1-KEY PRIVACY. We obtain 1-key privacy by returning to our observation about constrained evaluations. All constrained keys are essentially made up of PRF keys sampled from $\mathsf{PRF.Gen}$. Therefore, 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 4.4 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 simulation-based security [CC17], because we would have to permit $O(1)$ constrained key queries for the pseudorandomness requirement 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 $n_{\mathsf{in}} = n_{\mathsf{in}}(\kappa)$, and $n_{\mathsf{out}} = n_{\mathsf{out}}(\kappa)$ be integer-valued positive polynomials of the security parameter κ. A pseudorandom function is defined by a pair of PPT algorithms $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ where:

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

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

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 $x \in \{0,1\}^{n_{\mathsf{in}}}$ to the challenger, the challenger evaluates $y \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, x)$ and returns $y \in \{0,1\}^{n_{\mathsf{out}}}$ to A. It then updates $S \leftarrow S \cup \{x\}$.

Challenge Phase: At some point, A chooses its target input $x^\dagger \in \{0,1\}^{n_{\mathsf{in}}}$ such that $x^\dagger \notin S$ and submits it to the challenger. The challenger chooses a random coin $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. If $\mathsf{coin} = 0$, it evaluates $y^\dagger \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, x^\dagger)$. If $\mathsf{coin} = 1$, it samples a random value $y^\dagger \xleftarrow{\$} \{0,1\}^{n_{\mathsf{out}}}$. 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 $\widehat{\mathsf{coin}}$ as a guess for coin.

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

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 $n_{\mathsf{in}} = n_{\mathsf{in}}(\kappa)$, and $n_{\mathsf{out}} = n_{\mathsf{out}}(\kappa)$ 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 $\{0,1\}^{n_{\mathsf{in}}}$ 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 $x \in \{0,1\}^{n_{\mathsf{in}}}$ and outputs $y \in \{0,1\}^{n_{\mathsf{out}}}$.

$\mathsf{CPRF.Constrain}(\mathsf{K}, C) :\to \mathsf{K}_C$: 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 $x \in \{0,1\}^{n_{\mathsf{in}}}$ and outputs either $y \in \{0,1\}^{n_{\mathsf{out}}}$ or $\perp$.

Correctness. We define the notion of *correctness* for CPRFs. We say a CPRF Π_{CPRF} is correct if for all $\kappa \in \mathbb{N}$, $n_{\mathsf{in}}, n_{\mathsf{out}} \in \mathsf{poly}(\kappa)$, $\mathsf{K} \in \mathsf{CPRF.Gen}(1^\kappa)$, $C \in \mathcal{C}_\kappa$, $\mathsf{K}_C \in \mathsf{CPRF.Constrain}(\mathsf{K}, C)$, $x \in \{0,1\}^{n_{\mathsf{in}}}$ 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 $C : \{0,1\}^{n_{\mathsf{in}}} \to \{0,1\}^{n_{\mathsf{out}}}$, let $\mathsf{ConPoint} : \mathcal{C} \to \{0,1\}^{n_{\mathsf{in}}}$ 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 $x \in \{0,1\}^{n_{\mathsf{in}}}$, the challenger evaluates $y \leftarrow \mathsf{CPRF.Eval}(\mathsf{K}, x)$ and returns $y \in \{0,1\}^{n_{\mathsf{out}}}$ 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 $x^\dagger \in \{0,1\}^{n_{\mathsf{in}}}$ 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 coin $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. If $\mathsf{coin} = 0$, it evaluates $y^\dagger \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, x^\dagger)$. If $\mathsf{coin} = 1$, it samples a random value $y^\dagger \xleftarrow{\$} \{0,1\}^{n_{\mathsf{out}}}$. 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 $\widehat{\mathsf{coin}}$ as a guess for coin.

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

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). 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 satisfy only selective security for a subset of all possible queries. Constructions that achieve adaptive security are based on stronger assumptions (e.g. IO) 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*.

1-key privacy. We adopt the indistinguishability notion of 1-key privacy that was introduced by Boneh et al. [BLW17].[5] 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 coin $\mathsf{coin} \xleftarrow{\$} \{0,1\}$. The challenger then runs $\mathsf{K}_{\mathsf{coin}} \leftarrow \mathsf{CPRF.Constrain}(\mathsf{K}, C_{\mathsf{coin}})$ and returns $\mathsf{K}_{\mathsf{coin}}$ to A.

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

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

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.[6] 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 Constructing CPRFs 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 CPRFs where the constraints are the trivial singleton sets: $F = \{\{x\} \mid x \in \{0,1\}^{n_{\mathsf{in}}}\}$ or prefix-fixing predicates.

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

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

4.1 Preparation: Bit-Fixing Predicates

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

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

where v_i and x_i denotes the i^{th} bit of the string v and x, respectively. Then, the bit-fixing predicate (for length ℓ inputs) 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.

Moreover, 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} symbol of v. In addition, let G_{aut} be a function defined as

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

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

4.2 Construction

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

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

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

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

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

It then computes

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

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 $y \in \{0,1\}^{n_{\mathsf{out}}}$.

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

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

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

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

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

Finally, it outputs $y \in \{0,1\}^{n_{\mathsf{out}}}$.

4.3 Correctness

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

$$\mathsf{K}_v = \left(\widetilde{\mathsf{K}}_{T,w}\right)_{T\in[n_{\mathsf{in}}]^Q, w\in\{0,1\}^Q} \leftarrow \mathsf{CPRF.Constrain}(\mathsf{K}, C_v^{\mathsf{BF}}).$$

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

$$\bigwedge_{i=1}^{n_{\mathsf{in}}} \left((v_i \stackrel{?}{=} x_i) \bigvee (v_i \stackrel{?}{=} *)\right) = 1.$$

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

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

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

4.4 Pseudorandomness on Constrained Points

Theorem 4.1. *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.*

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 $\widehat{\mathsf{coin}} = \mathsf{coin}$ 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 $x \in \{0,1\}^{n_{\mathsf{in}}}$ 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 $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_{n_{\mathsf{in}}}$ as the j^{th} ($j \in [Q]$) constrained key query, the challenger returns $\mathsf{K}_{v^{(j)}} \leftarrow \mathsf{CPRF.Constrain}(\mathsf{K}, C^{\mathsf{BF}}_{v^{(j)}})$ to A and updates $S_{\mathsf{con}} \leftarrow S_{\mathsf{con}} \cup \{C^{\mathsf{BF}}_{v^{(j)}}\}$.

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

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

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

Game_1**:** In this game, we add an extra abort condition for the challenger. Specifically, at the end of the game, the challenger samples a random set $T^\dagger \xleftarrow{\$} [n_{\mathsf{in}}]^Q$. Let us set $T^\dagger = (t_1, \cdots, t_Q)$. The challenger further samples $b^\dagger_{t_j} \xleftarrow{\$} \{0,1\}$ for all $j \in [Q]$. Let $b^\dagger_{T^\dagger} := b_{t_1} b_{t_2} \cdots b_{t_Q} \in \{0,1\}^Q$. 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
$$\left((v^{(j)}_{t_j} \stackrel{?}{=} b^\dagger_{t_j}) \bigvee (v^{(j)}_{t_j} \stackrel{?}{=} *) \right) = 0. \tag{2}$$
does not hold.
- The challenger aborts if $x^\dagger$ does not satisfy
$$(b^\dagger_{T^\dagger} \stackrel{?}{=} x^\dagger_{T^\dagger}) = \bigwedge_{j \in [Q]} (b^\dagger_{t_j} \stackrel{?}{=} x^\dagger_{t_j}) = 1. \tag{3}$$
- 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 $[n_{\mathsf{in}}]^Q \times \{0,1\}^Q$ such as the lexicographic order) that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). Note that it is possible to efficiently find such a pair by enumerating over $[n_{\mathsf{in}}]^Q \times \{0,1\}^Q$ since $Q = O(1)$.[7]

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

As we will show in Lemma 4.2, there exists at least a single pair $(T^\dagger, b^\dagger_{T^\dagger}) \in [n_{\mathsf{in}}]^Q \times \{0,1\}^Q$ that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). Therefore, the event abort occurs with probability $1 - 1/(2n)^Q$. Furthermore, it can be seen that abort occurs independently from the view of A. Therefore, we have

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

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

[7] One may wonder why the final condition for the abort is necessary, because the reduction in the proof of Lemma 4.3 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.

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 $C^{\mathsf{BF}}_{v(j)} \in \mathcal{C}^{\mathsf{BF}}_{n_{\mathsf{in}}}$ 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 it is only a conceptual change, we have

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

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

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

using the master secret key

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

that it constructed at the beginning of the game, where $\{\overline{\mathsf{K}}_{T,w}, \widehat{\mathsf{K}}_{T,w}\} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [n_{\mathsf{in}}]^Q$ 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

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

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

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

We show in Lemma 4.3 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 $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ are exactly the same since A has not made an evaluation query on $x^\dagger$ and $\overline{\mathsf{K}}_{T^\dagger, b^\dagger_{T^\dagger}}$ is not given through any of the constrained key query. Concretely, $\bar{y}^\dagger$ is distributed uniform random regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$ and thus the value of coin is information theoretically hidden to A. Therefore, we have

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

Combining everything together with Lemma 4.2 and Lemma 4.3, we have

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

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

Lemma 4.2. *In* Game_1*, we have*

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

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

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

Let us denote $\bar{T} := (t^{(1)}, \cdots, t^{(Q)}) \in [n_{\mathsf{in}}]^Q$ 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 4.3. *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 $T^\dagger = (t_1, \cdots, t_Q) \xleftarrow{\$} [n_{\mathsf{in}}]^Q$ 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 $\overline{\mathsf{K}}_{T,w} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [n_{\mathsf{in}}]^Q$ 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

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

Here, B implicitly sets $\overline{\mathsf{K}}_{T^\dagger, b^\dagger_{T^\dagger}}$ 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 $x \in \{0,1\}^{n_{\mathsf{in}}}$ 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

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

  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

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

  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 $\bar{y} = \mathsf{PRF.Eval}(\overline{\mathsf{K}}_{T^\dagger, b^\dagger_{T^\dagger}}, x)$.

- When A submits $C^{\mathsf{BF}}_{v^{(j)}} \in \mathcal{C}^{\mathsf{BF}}_{n_{\mathsf{in}}}$ 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:

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

where $\widetilde{\mathsf{K}}_{T,w} = \overline{\mathsf{K}}_{T,w}$ if and only if $w \in G_{\mathsf{aut}}(v_T^{(j)}) = \{w \in \{0,1\}^Q \mid C^{\mathsf{BF}}_{v_T^{(j)}}(w) = 1\}$ 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 $b^\dagger_{T^\dagger} \notin G_{\mathsf{aut}}(v^{(j)}_{T^\dagger})$, 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 $\overline{\mathsf{K}}_{T^\dagger,b^\dagger_{T^\dagger}}$ is not included in $\mathsf{K}_{v^{(j)}}$.

- When A submits the target input $x^\dagger \in \{0,1\}^{n_{\mathsf{in}}}$ 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, $\overline{\mathsf{K}}_{T^\dagger,b^\dagger_{T^\dagger}}$ must be required to compute on input $x^\dagger$.

Finally, A outputs its guess $\widehat{\mathsf{coin}}$. 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 $\widehat{\mathsf{coin}}$ as its guess.

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

This completes the proof. □

Theorem 4.4. *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 secret key is of the form:

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

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

- $\left(\widetilde{\mathsf{K}}^{(0)}_{T,w}\right)_{T\in[n_{\mathsf{in}}]^Q, w\in\{0,1\}^Q}$ where $\widetilde{\mathsf{K}}^{(0)}_{T,w} = \overline{\mathsf{K}}_{T,w}$ if and only if $w \in G_{\mathsf{aut}}(v^{(0)}_T)$, and $\widetilde{\mathsf{K}}_{T,w} = \widehat{\mathsf{K}}_{T,w}$ otherwise.
- $\left(\widetilde{\mathsf{K}}^{(1)}_{T,w}\right)_{T\in[n_{\mathsf{in}}]^Q, w\in\{0,1\}^Q}$ where $\widetilde{\mathsf{K}}^{(1)}_{T,w} = \overline{\mathsf{K}}_{T,w}$ if and only if $w \in G_{\mathsf{aut}}(v^{(1)}_T)$, 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. □

5 Conclusion and future work

In conclusion, we have developed the first CPRF construction with $O(1)$ collusion-resistance for bit-fixing predicates. Our technique signals a noted departure from existing techniques and uses much weaker assumption than any previous constructions for comparable predicates. Finally we achieve full

adaptive security with a polynomial security loss alongside 1-key privacy. Our construction is the first to achieve adaptive security in the standard model and without sub-exponential security losses (regardless of the primitive).

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 constant collusion-resistant CPRFs for bounded-depth circuits.
- Devise a construction that satisfies $O(p(\kappa))$ collusion-resistance for some possibly bounded polynomial p in κ for BF predicates.
- Satisfy key privacy for > 1 constrained keys for BF 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 even though 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], CPRF for r-key privacy for *BF predicates* may be much easier to construct. It merely implies IO for BF predicates, which is trivial to construct since the BF predicates have its efficiently computable standard form. We also remark that distributional VBB obfuscation for BF predicates is a different thing from IO for BF 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

[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 2, 3, 4.)

[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, 4.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012. (Cited on page 3.)

[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, 4.)

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Kalai and Reyzin [KR17], pages 567–594. (Cited on page 4.)

[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 5.)

[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 17.)

[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, 8, 10.)

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012. (Cited on page 5.)

[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.)

[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, 4.)

[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 17.)

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

[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, 8, 10, 17.)

[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.)

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, March 2015. (Cited on page 17, 18.)

[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.)

[FOC17] *58th FOCS*. IEEE Computer Society Press, 2017. (Cited on page 19.)

[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, 8, 10.)

[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 4.)

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In FOCS 2017 [FOC17], pages 612–621. (Cited on page 17.)

[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 8, 10.)

[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, 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.)

[KR17] Yael Kalai and Leonid Reyzin, editors. *TCC 2017, Part II*, volume 10678 of *LNCS*. Springer, Heidelberg, November 2017. (Cited on page 17, 18.)

[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.)

[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.)

[SB18] Hovav Shacham and Alexandra Boldyreva, editors. *CRYPTO 2018, Part II*, volume 10992 of *LNCS*. Springer, Heidelberg, August 2018. (Cited on page 17, 18.)

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005. (Cited on page 7, 13.)

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In FOCS 2017 [FOC17], pages 600–611. (Cited on page 17.)