Note on Constructing Constrained PRFs from OWFs with Constant Collusion Resistance

Shuichi Katsumata [*] Shota Yamada [†]

September 26, 2018

Abstract

Constrained pseudorandom functions (CPRFs) are a type of PRFs that allows one to derive a *constrained* key K_C from the master key K. While the master key K allows one to evaluate on any input as a standard PRF, the constrained key K_C only allows one to evaluate on inputs x such that $C(x) = 1$. Since the introduction of CPRFs by Boneh and Waters (ASIACRYPT'13), Kiayias et al. (CCS'13), and Boyle et al. (PKC'14), there have been various constructions of CPRFs. However, thus far, almost all constructions (from standard assumptions and non-trivial constraints) are only proven to be secure if at most one constrained key K_C is known to the adversary, excluding the very recent work of Davidson and Nishimaki (EPRINT'18). Considering the interesting applications of CPRFs such as ID-based non-interactive key exchange, we desire CPRFs that are collusion resistance with respect to the constrained keys. In this work, we make progress in this direction and construct a CPRF for the bit-fixing predicates that are collusion resistance for a constant number of constrained keys. Surprisingly, compared to the heavy machinery that was used by previous CPRF constructions, our construction only relies on the existence of one-way functions.

1 Introduction

Constrained pseudorandom functions (CPRFs), introduced simultaneously by Boneh and Waters [BW13], Kiayias et at. [KPTZ13], and Boyle et al. [BGI14], are a generalization of standard pseudorandom functions (PRF) [GGM84] which allows one to derive a *constrained* key that can be used to evaluate the PRF on a subset of the inputs satisfied by the constraint. So far there have been numerous constructions of CPRFs from various assumptions [BW13, HKKW14, BV15, BLW17, CC17, BTVW17, PS18, CVW18, AMN+18, DN18]. However, beside the CPRFs for trivial constraints such as constraints represented by singleton sets: $F = \{\{x\} \mid x \in \{0,1\}^{n_{in}}\}$ or prefix-fixing predicates which are satisfied by the classical Goldreich-Goldwasser-Micali PRF [GGM84], much is left to be desired. For instance, the constructions that satisfy full *collusion-resistance* of the constrained keys, i.e., pseudorandomness can be argued even when the adversary obtains polynomially many constrained keys that are constrained on the challenge points, all rely on the random oracle model or assume the existence of indistinguishable obfuscation or multi-linear maps [BW13, HKKW14, BLW17]. Very recently, [DN18] made great progress in this direction and showed a CPRF construction for the bit-fixing predicate that achieves constant key

[*]The University of Tokyo, National Institute of Advanced Industrial Science and Technology (AIST). E-mail: `shuichi_katsumata@it.k.u-tokyo.ac.jp`

[†]National Institute of Advanced Industrial Science and Technology (AIST). E-mail: `yamada-shota@aist.go.jp`

collusion-resistance based on standard lattice-based assumptions in the standard model. Considering some of the interesting applications of CPRFs such as non-interactive key exchange [BLW17], it is desirable to have CPRFs that are secure even if the adversary is allowed to obtain multiple constrained keys. Moreover, other than [HKKW14, DN18], all constructions are only proven to be selectively-secure, where the adversary must commit to the constraint of the constrained key it receives from the challenger at the beginning of the game.

In this paper, we provide an *adaptive* and *collusion-resistant* CPRF for the bit-fixing predicate, where the scheme can tolerate up to any *constant* Q-collusions of constrained keys. Our construction achieves the same security requirement as [DN18]. Compared to previous CPRFs that required heavy machinery, our construction is very simple and can be proven solely from the existence of one-way functions.

2 Preliminaries

2.1 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 $x \in \{0,1\}^{n_{in}}$ and outputs $y \in \{0,1\}^{n_{out}}$.

Pseudorandomness. We define the notion of (adaptive) *pseudorandomness* for PRFs $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$. The 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 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 $x \in \{0,1\}^{n_{in}}$ to the challenger, the challenger evaluates $y \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, x)$ and returns $y \in \{0,1\}^{n_{out}}$ to A. It then updates $S \leftarrow S \cup \{x\}$.

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

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

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 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 are implied from one-way functions [GGM84, HILL99].

2.2 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 $\{0,1\}^{n_{in}}$ 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 $x \in \{0,1\}^{n_{in}}$ and outputs $y \in \{0,1\}^{n_{out}}$.

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

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

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

Guess: Eventually, 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 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 1 (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.*

Remark 2 (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*.*

3 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. CPRFs with the same property were very recently constructed by Davidson and Nishimaki [DN18], however, their results require the LWE assumption. Other than ours and [DN18], 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: $F = \{\{x\} \mid x \in \{0,1\}^{n_{in}}\}$ or prefix-fixing predicates [GGM84, BW13, BLW17].

3.1 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

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

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 $Q \leq \ell$, let us define $v_T \in \{0,1,*\}^Q$ as the string $v_{t_1} v_{t_2} \cdots v_{t_Q}$, where v_i is the i-th 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 cared 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.

3.2 Construction

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

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

$\mathsf{CPRF.Eval}(\mathsf{K}, x) :\to y$: On input the master key K and input $x \in \{0,1\}^{n_{\mathsf{in}}}$, it first parses $\left(\bar{\mathsf{K}}_{T,w}\right)_{T \in [n_{\mathsf{in}}]^Q, w \in \{0,1\}^Q} \leftarrow \mathsf{K}$. It then computes

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

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

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

$$\mathsf{K}_v = \left(\left(\bar{\mathsf{K}}_{T,w}\right)_{w \in G_{\mathsf{aut}}(v_T)}\right)_{T \in [n_{\mathsf{in}}]^Q},$$

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

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

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

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

3.3 Correctness

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

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

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

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

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

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

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

3.4 Pseudorandomness on Constrained Points

We show the following lemma.

Theorem 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 $x \in \{0,1\}^{n_{in}}$ 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 $C_{v^{(j)}}^{\mathsf{BF}} \in \mathcal{C}_{n_{in}}^{\mathsf{BF}}$ 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 $x^* \in \{0,1\}^{n_{in}}$ as the challenge query, we have the restriction $x^* \notin S_{\mathsf{eval}}$ and $x^* \in \mathsf{ConPoint}(C_{v^{(j)}}^{\mathsf{BF}})$ for all $C_{v^{(j)}}^{\mathsf{BF}} \in S_{\mathsf{con}}$. Here, the latter condition is equivalent to

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

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

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

$$(v^{(j)}_{t_j} \overset{?}{=} b^*_{t_j}) \bigvee (v^{(j)}_{t_j} \overset{?}{=} *) = 0. \tag{2}$$

  does not hold.

- The challenger aborts if x^* does not satisfy

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

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

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 1, there exists at least single pair $(T^*, b^*_{T^*}) \in [n_{\mathsf{in}}]^Q \times \{0,1\}^Q$ 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.

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

[1] One may wonder why the final condition for the abort is necessary, because the reduction in the proof of Lemma 2 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.

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 $x^* \in \{0,1\}^{n_{\mathsf{in}}}$ 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 $\mathsf{CPRF.Eval}(\mathsf{K}, x^*)$ as

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

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

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

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

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

We show in Lemma 2 that

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

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

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

Combining everything together with Lemma 1 and Lemma 2, we have

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

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

Lemma 1. *In* Game_1*, we have*

$$\left\{(T^*, b^*_{T^*}) \in [n_{\mathsf{in}}]^Q \times \{0,1\}^Q \mid (T^*, b^*_{T^*}) \textit{ satisfies Equation (2) for all } j \in [Q] \textit{ and Equation (3)}\right\} \ne \emptyset.$$

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

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

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

Lemma 2. *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 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 $T^* = (t_1, \cdots, t_Q) \xleftarrow{\$} [n_{\mathsf{in}}]^Q$ and $b^*_{t_j} \xleftarrow{\$} \{0,1\}$ for all $j \in [Q]$ as in the Game_2-challenger. Let $b^*_{T^*} := b_{t_1} b_{t_2} \cdots b_{t_Q} \in \{0,1\}^Q$. Then, it further samples $\bar{\mathsf{K}}_{T,w} \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ for all $T \in [n_{\mathsf{in}}]^Q$ and $w \in \{0,1\}^Q$ *except* for $(T^*, b^*_{T^*})$. It then sets the (simulated) master key K^* as

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

Here, B implicitly sets $\bar{\mathsf{K}}_{T^*, b^*_{T^*}}$ 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 $x \in \{0,1\}^{n_{in}}$ as the evaluation query, B checks whether $x_{T^*} = b^*_{T^*}$. If not, then it can use the simulated master key K^* to compute

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

  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

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

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

- 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:

$$\mathsf{K}_{v^j} = \left(\left(\bar{\mathsf{K}}_{T,w} \right)_{w \in G_{\mathsf{aut}}(v^{(j)}_T)} \right)_{T \in [n_{\mathsf{in}}]^Q},$$

  where recall $G_{\mathsf{aut}}(v^{(j)}_T) = \{ w \in \{0,1\}^Q \mid C^{\mathsf{BF}}_{v^{(j)}_T}(w) = 1 \}$. Here, B can prepare all the PRF keys since condition in Equation (2) guarantees us that we have $b^*_{T^*} \notin G_{\mathsf{aut}}(v^{(j)}_{T^*})$, or equivalently, $C^{\mathsf{BF}}_{v^{(j)}_{T^*}}(b^*_{T^*}) = 0$. Namely, $\bar{\mathsf{K}}_{T^*, b^*_{T^*}}$ is not included in $\mathsf{K}_{v^{(j)}}$.

- When A submits the target input $x^* \in \{0,1\}^{n_{in}}$ 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^* as its challenge query and receives back $\bar{y}^*$. It then computes y^* as in Equation (6) and returns y^* to A. Here, since Equation (3) holds, $\bar{\mathsf{K}}_{T^*, b^*_{T^*}}$ must be required to compute on input x^*.

Finally, A outputs its guess $\widehat{\mathsf{coin}}$. B then checks whether $(T^*, b^*_{T^*})$ is the first pair that satisfies Equation (2) for all $j \in [Q]$ and Equation (3). If it does not hold, 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 $\bar{y}^* \leftarrow \mathsf{PRF.Eval}(\bar{\mathsf{K}}_{T^*, b^*_{T^*}}, x^*)$, hence B simulates Game_2 perfectly. Otherwise in case $\mathsf{coin} = 1$, B receives $\bar{y}^* \xleftarrow{\$} \{0,1\}^{n_{out}}$, 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. $\square$

This completes the proof. $\square$

References

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

[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. 1

[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. 1, 2, 4

[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. 1

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

[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. 1, 4

[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. 1

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018. 1

[DN18] Alex Davidson and Ryo Nishimaki. A bit-fixing prf with $O(1)$ collision-resistance from lwe. Cryptology ePrint Archive, Report 2018/890, 2018. https://eprint.iacr.org/2018/890. 1, 2, 4

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984. 1, 3, 4

[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. 3

[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. 1, 2

[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. 1

[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. 1

[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. 7