Constrained Pseudorandom Functions from Pseudorandom Synthesizers

Zachary A. Kissel[†]

[†]Department of Computer Science, Merrimack College
kisselz@merrimack.edu

Abstract

In this paper we resolve the question of whether or not constrained pseudorandom functions (CPRFs) can be built directly from pseudorandom synthesizers. In particular, we demonstrate that the generic PRF construction from pseudorandom synthesizers due to Naor and Reingold can be used to construct CPRFs with bit-fixed predicates using the "direct-line" approach. We further introduce a property of CPRFs that may be of independent interest.

1 Introduction

A pseudorandom function (PRF) is a family of indexed (keyed) functions $\{f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ such that the behavior of a function selected uniformly at random from the indexed family is indistinguishable, by a polynomial time adversary, from the behavior of a function chosen randomly from the set of all functions $\{f : \mathcal{X} \to \mathcal{Y}\}$ when given only oracle access to the function. It has been shown that this primitive can be constructed, in a black-box way, from length-doubling pseudorandom generators (PRGs) [GGM86] and from pseudorandom synthesizers [NR95, NR04]. A pseudorandom synthesizer can be intuitively thought of as a length squaring PRG. Both constructions use a binary tree to determine the output of the function. In the case of [GGM86] we determine the value using a root to leaf path where as in [NR95] we determine the value by using a leaf to root level order evaluation.

A recent extension to PRFs, known as a *constrained pseudorandom function* (CPRF), allow a party that knows the key (also called the *master key*) to a PRF to delegate evaluation to a third party by providing a special *constrained key* with an associated evaluation predicate. The delegatee can use the constrained key to evaluate the PRF at a point $x \in \mathcal{X}$ if x satisfies the predicate; otherwise the output will appear random (or possibly be undefined) [BW13, KPTZ13, BGI14]. GGM based CPRF constructions are built from the following core observation: a GGM PRF can be evaluated by a third party that has access to some collection of values associated with nodes in the tree (which form the constrained key).

Given CPRFs exist from the GGM PRF construction, it is natural to ask the question:

Can constrained PRFs be built from pseudorandom synthesizers using the PRF construction of [NR95]?

Our work answers this question in the affirmative.

Construction Overview. Briefly, the PRF from [NR95] takes ℓ-bit inputs, where ℓ is a power of two, and maps them to n-bit values using a (master) key that is a $2 \times \ell$ bit matrix of random n-bit values. The first row contains n-bit strings representing the zero values associated with all ℓ bits. Similarly, the second row represents the one values. To evaluate the PRF, a binary tree is constructed where each leaf is given the entry of master key associated with the bit. The internal nodes become the result of the synthesizer $S : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ applied to its two children. Our main idea is that given the master key we can construct a constrained key by building a new $2 \times \ell$ matrix where some entries come from the master key while other entries are randomly sampled from $\{0,1\}^n$. The evaluation using the constrained key is similar to the evaluation with the master key (except for the matrix from which the values are drawn). This will allow us to realize a predicate known as *bit-fixing*, the input $x \in \{0,1\}^n$ bit-wise matches a pattern string $p \in \{0,1,*\}^n$ where $*$ denotes a wildcard.

Comparison to [GGM86]-Based CPRFs. While the GGM based CPRF of [KPTZ13] supports range predicates (the input $x \in \mathcal{X}$ falls within a certain range), our pseudorandom synthesizer based CPRF supports bit fixing, a more general predicate that implies range predicates. Both the GGM and pseudorandom synthesizer based CPRF are selectively secure meaning the constrained key predicates must be declared *before* interrogating the evaluation oracle. Additionally, our construction limits the number of constrained key predicates the adversary can obtain to one. Both constructions support unbounded delegation by which we mean a party that posses a constrained key for some predicate can construct a key for a more restrictive predicate *without* the PRF master key. Lastly in contrast to the generic constructions from the GGM PRF, our synthesizer based CPRF is inherently parallel with a shorter evaluation tree (logarithmic in the size of the input) and thus, more efficient than the constructions based on the GGM PRF (linear in the size of the input). It is important to recall that the PRF of [NR95] and [GGM86] treat their building blocks as a black box therefore, richer CPRF constructions are possible using non-blackbox techniques.

1.1 Related Work

The work in constrained pseudorandom function can be divided in to roughly three main areas: (1) constructing CPRFs for interesting predicates; (2) considering more advanced notions of security, in particular predicate privacy; or (3) applying CPRFs to existing problems.

To point one, constructions have been discovered for many predicates. We survey them here.

Puncturing. A predicate that evaluates to true for all $x \in \mathcal{X} \setminus \{x^*\}$. The constructions of [BW13] and [KPTZ13] realize puncturing from the PRF construction of [GGM86]. Together with $i\mathcal{O}$, punctured PRFs give rise to the *punctured programming approach* [SW21].

Ranges. A predicate that evaluates to true for all $x \in \mathcal{X}$ that fall within a certain range. One of the original construction to introduce the idea of CPRFs [KPTZ13], demonstrated how to realize the predicate using the GGM PRF as a foundation.

Prefix. A predicate that evaluates to true for all $x \in \mathcal{X}$ that share some common prefix. This predicate can be realized using the GGM PRF which was was done directly by [BW13] and [BGI14]. The construction of [KPTZ13] also allows for prefix predicate since a prefix is a special case of their range query construction.

Left-Right. Consider a PRF $f : \mathcal{K} \times \mathcal{X}^2 \to \mathcal{Y}$ and the fixed value $w \in \mathcal{X}$, the predicate will return true if either input $(x, y) = (w, y)$ or (x, w). Unlike other predicates, this requires a constrained key pair $k_{w,\text{left}}$ and $k_{w,\text{right}}$. This predicate was first considered by [BW13] and realized using a bilinear map based construction in the random oracle model.

Bit-Fixing. A predicate that evaluates to true if $x \in \{0,1\}^n$ matches a pattern string from $p = \{0, 1, *\}^n$ where a match is defined as $\bigwedge_{i=1}^{n} \left(\left(p_i \overset{?}{=} x_i \right) \vee \left(p_i \overset{?}{=} * \right) \right)$. It should be noted that a bit fixing predicate is more expressive than it may look at first; it captures prefix, suffix (true for all $x \in \mathcal{X}$ such that x ends with s), range, and left-right predicates. The construction has been realized using κ-linear maps by [BW13] applied to a PRF that is very similar to the PRF from [NR04]. More recently, [DKN$^+$20] were able to realize bit-fixing predicates by modifying the simple distributed PRF of [MS95], defined as: $f_{k_1,\ldots,k_n}(x) = \bigoplus_{i=1}^{n} \hat{f}_{k_i}(x)$, where $\hat{f}$ is some PRF. In particular, they showed that if only Q constrained keys were generated by the adversary, collusion using keys is impossible. The drawback is the the number of keys used in the construction is on the order of $(2n)^Q$ where $Q \ll n$ is a constant; otherwise, the size of the key would be impractical.

Circuits. A predicate that evaluates to true for all $x \in \mathcal{X}$ if a circuit C outputs one when given x. While most constructions do not offer all possible circuits: [BW13] use multilinear maps to support *monotone* circuits, [CC17] use the LWE assumption to support NC^1 while [AMN$^+$18] use correlated input hash functions to support the same class. The work of [DKN$^+$20] constructs a predicate for the conjunction of NC^0 circuits (t-CNF) from one-way functions, a similar construction was also found by [Tsa19]. Allowing for the use of $i\mathcal{O}$, the work of [HKKW19] supports generic circuits and [DKN$^+$20] supports circuits in $\mathsf{P/poly}$.

To point two, many works have considered the notion of predicate privacy [BTVW17, BLW17, BKM17, CC17, AMN$^+$18, PS20, ...]. In these CPRF families the constrained key does not reveal its associated predicate. Constructions that hide constraints allow for the construction of deniable encryption and watermarking, among other applications.

Lastly, the application space for constrained PRFs is rich, mainly due to the fundamental nature of PRFs. Some highlights include: deniable encryption [BLW17], functional signatures [BGI14], key regression [Kis19], searchable symmetric encryption (SSE) [KPTZ13], e-Cash [BPS19], and identity-based non-interactive key exchange (ID-NIKE) [BW13].

2 Preliminaries

In what follows we will denote the security parameter by λ and the set of binary strings of length n by $\{0,1\}^n$. We define a predicate family Φ as a set of predicates $\varphi : \mathcal{X} \to \{0,1\}$. We denote matrices by upper case letters in a san serif font (e.g., K) and the elements of the matrix will be represented by the lower-case version of the letter, appropriately indexed. We denote the set $\{1,2,\ldots,n\}$ by $[n]$, and an arbitrary negligible function by $\mathsf{negl}(\cdot)$. We denote by $a \xleftarrow{\$} S$ assigning a a uniformly random sample from set S. We will use the notation $D_1 \approx_c D_2$ to denote that two distributions are *computationally* indistinguishable.

2.1 Constrained Pseudorandom Functions

A constrained pseudorandom function (CPRF) is a tuple of algorithms
$\Pi = (\mathsf{F.Gen}, \mathsf{F.Constrain}, \mathsf{F.Eval}, \mathsf{F.ConstrainedEval})$ defined over domain $\mathcal{X}$ and range $\mathcal{Y}$ with the following properties:

- $\mathsf{F.Gen}\left(1^\lambda\right)$ takes as input a security parameter, 1^λ and samples a random function from PRF family by sampling a master key msk from the appropriate keyspace.
- $\mathsf{F.Constrain}\left(\mathsf{msk}, \varphi\right)$takes the master key and predicate φ as input and returns a key k_φ that represents the constrained key.
- $\mathsf{F.Eval}\left(\mathsf{msk}, x\right)$ takes the master key and an input $x \in \mathcal{X}$ and returns the appropriate value $y \in \mathcal{Y}$.
- $\mathsf{F.ConstrainedEval}\left(k_\varphi, x\right)$ takes as input a constrained key k_φ and an $x \in \mathcal{X}$. If $\varphi(x) = 1$ the correct value $y \in \mathcal{Y}$ is returned (i.e. $\mathsf{Eval}\left(k_\varphi, x\right)$); otherwise either a random value in $\mathcal{Y}$ or $\perp$ is returned.

Correctness. We say a CPRF is *correct* if the evaluation of x using the constrained key agrees with the evaluation of x using the master key when the predicate associated with the constrained key is satisfied. More formally, F is *correct* if for all $x \in \mathcal{X}$ where $\varphi(x) = 1$, master keys msk, and constrained keys k_φ, $\mathsf{F.Eval}(\mathsf{msk}, x) = \mathsf{F.ConstrainedEval}(k_\varphi, x)$.

Security Notion. A constrained pseudorandom function has a nuanced security guarantee. With a traditional PRF, security states that under a polynomial number of adaptive queries a probabilistic-polynomial time (PPT) adversary can not distinguish between the output of a PRF and the output of a randomly selected function of the same domain and range. When it comes to CPRFs security is relative to a predicate $\varphi \in \Phi$ where the adversary will be able to evaluate the PRF at any point $x \in \mathcal{X}$ where $\varphi(x) = 1$ without the assistance of an evaluation oracle.

Given a CPRF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we can cast the security notion as a game $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{CPRF-Secure}}(b, \lambda)$ where $\mathcal{A}$ is a probabilistic polynomial-time adversary, $b \in \{0,1\}$, and λ is a security parameter we define the security game $\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}(b, \lambda)$ as follows:

$\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}(b, \lambda)$:

1. The game runs $\mathsf{F.Gen}(1^{\lambda})$ to obtain master key msk.
2. If $b = 0$, the game samples a random function $R : \mathcal{X} \to \mathcal{Y}$.
3. The adversary is provided access to two oracles:
   - (a) A $\mathsf{F.Constrain}$ oracle which allows $\mathcal{A}$ to ask the game for a constrained key k_{φ} for predicate $\varphi \in \Phi$, adding φ to P, the set of queried predicates.
   - (b) An $\mathsf{F.Eval}$ oracle which behaves as follows on query x:
     - If $b = 0$, answer the evaluation queries as follows: if $\varphi(x) = 1$ for some $\varphi \in P$, respond honestly. If there is no such predicate, output the value of $R(x)$.
     - If $b = 1$, return the output of $\mathsf{Eval}(\mathsf{msk}, x)$.
4. Eventually the $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

We say that a CPRF is secure if for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$\left|\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}(0, \lambda) = 1\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}(1, \lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Remark 2.1. If we modify the CPRF-secure game so we force the adversary to commit to some challenge point *before* they get access to the oracle we arrive at the (challenge) selective security game.

Remark 2.2. If in the game the adversary is not given access to a constrain oracle but, rather can ask for exactly one constrained key *before* being given access to the the evaluation oracle, we say that the security notion is (constrained-key) selective. This notion of security was (seemingly first) considered by [BV15] and we will use the notion in this work.

Predicate Privacy. Roughly speaking, predicate privacy implies that the party that posses the constrained key can not learn the associated predicate. We can formalize this notion as a game $\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\text{CPRF}-\text{Priv}}(\lambda)$ (first defined by [BTVW17]) between a challenger and a PPT adversary $\mathcal{A}$ as follows:

$\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\text{CPRF}-\text{Priv}}(\lambda)$:

1. Challenger runs $\mathsf{msk} \leftarrow \mathsf{F.Gen}\left(1^{\lambda}\right)$ and constructs an empty set of queries Q.
2. Adversary $\mathcal{A}$ is given access to an evaluation oracle that on query x returns $y \leftarrow \mathsf{F.Eval}(\mathsf{msk}, x)$ to $\mathcal{A}$. Every query x is added to Q.
3. Eventually, $\mathcal{A}$ outputs two predicates φ_0 and φ_1 from Φ such that for all $x \in Q$, $\varphi_0(x) = \varphi_1(x)$. The challenger samples a bit $b \xleftarrow{\$} \{0,1\}$ and returns $k_{\varphi_b} \leftarrow \mathsf{F.Constrain}(\mathsf{msk}, \varphi_b)$ to $\mathcal{A}$.
4. $\mathcal{A}$ is given access to the evaluation oracle but, is not allowed to query on a value of x where $\varphi_0(x) \neq \varphi_1(x)$; otherwise, $\mathcal{A}$ can trivially distinguish between the two keys.
5. Eventually, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. The adversary $\mathcal{A}$ wins the game if $b = b'$.

We say that a CPRF F is one-key private if for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\text{CPRF}-\text{Priv}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

The game, as presented, is known as *one-key privacy* as the adversary is only allowed to make one challenge query. The security notion can be extended to d-key privacy by allowing the adversary to issue d pairs of predicates. In the case of d-key privacy, what is admissible for the adversary must be updated, see [BLW17] for details.

2.2 The Direct-Line Approach

As highlighted in [DKN+20] many bit-fixed CPRF constructions (e.g., [DKN+20, CC17, ...]) use a so called *direct-line* approach. In the simplest form of this approach, each bit position i of an ℓ-bit input PRF is assigned one of two possible random values, one that represents zero $r_{0,i}$ and one that represents one $r_{1,i}$. During evaluation of an input string x, the random values $r_{x_0,0}, \ldots, r_{x_{\ell-1},\ell-1}$ are combined in some way to arrive at the output of the PRF. The key of the PRF, K is always a $2 \times \ell$ matrix of random values where the top row represents zeros and the bottom row represents ones. To bit-fix according to a pattern p, a key K_{φ_p} is constructed such that

$$k_{\varphi_p,i,j} = \begin{cases} k_{\varphi_p i,j} = k_{i,j} & \text{if } p_j = * \text{ or } p_j = i, \\ r_{i,j} \xleftarrow{\$} \{0,1\}^n & \text{otherwise.} \end{cases}.$$

In practice the random values that are assigned to elements of K_{φ_p} are committed to in the CPRF Gen algorithm. A natural extension of the direct-line approach is to divide the input into blocks of bits and follow a similar approach.

We note that, as presented, the direct-line approach is 1-key private and is secure provided the adversary only obtains access to one constrained key. To see this observe that every key matrix is simply a uniformly random matrix and thus is indistinguishable from any other uniformly random matrix.

In [CC17] these random values are random matrices (making the key matrix a tensor). The output of the PRF is a rounded product of the appropriate random matrices with an additional randomly chosen matrix, resembling the PRF of [NR04]. The security of their construction relies on the LWE assumption.

The work of [DKN$^+$20] demonstrated that the direct line approach can be applied using the distributed XOR PRF [MS95]. In particular, they treat the random values as keys to individual sub-PRFs which are appropriately combined using the exclusive-or of the evaluation of the sub-PRFs on the desired input $x \in \{0,1\}^\ell$. Specifically, $F_{\mathsf{K}}(x) = \bigoplus_{i=0}^{\ell-1} \hat{f}_{r_{x_i,i}}(x)$, where $\hat{f}_{r_{x_i,i}}$ is a PRF with key $r_{x_i,i}$. They further extended the direct-line approach so that a constant number of constrained keys can be obtained by the adversary while simultaneously maintaining security at the cost of polynomially larger keys.

Additionally in the work of [DKN$^+$20], it was shown that the direct line approach can be applied to a t-CNF circuit which is equivalent to the conjunction of NC^0 circuits where each circuit only has $t < \ell$ bits of input. Though it should be noted that this is representing blocks of bits at a time with no requirement that these blocks be constructed from adjacent bits.

In this work we use the direct line approach to realize bit-fixing CPRFs from pseudorandom synthesizers.

2.3 Pseudorandom Synthesizers

A pseudorandom synthesizer [NR95] is a family of polynomial-time computable functions $\{S : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n\}$ where for a specific S and two random sets of q values $X = \{x_1, x_2, \ldots, x_q\}$ and $Y = \{y_1, y_2, \ldots, y_q\}$ where $x_i, y_i \in \{0,1\}^n$, the q^2 values $S(x_i, y_j)$ are indistinguishable from q^2 random strings drawn uniformly from $\{0,1\}^n$. More formally, we say that the matrix of values generated by applying the synthesizer to every possible pair of elements from X and Y, denoted $C_S(X,Y)$, is computationally indistinguishable from a uniformly random $q \times q$ matrix of n-bit strings. It is known, from [NR95], that pseudorandom synthesizers can be constructed from *weak* PRFs[1] and thus one-way functions.

Given a pseudorandom synthesizer S one can construct a pseudorandom function family $\left\{f : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^n\right\}$ where ℓ is a power of two. The PRF family is indexed by a $2 \times \ell$ matrix with elements drawn from $\{0,1\}^n$. The top row of the matrix represents the binary strings associated with zeros and the bottom row of the matrix represents the binary

[1] A PRF $f_k : \mathcal{X} \to \mathcal{Y}$ where $\{(x_i, f_k(x_i))\}_{i=1}^n \approx_c \{(x_i, y_i)\}_{i=1}^n$.

strings associated with ones. Each column of the matrix corresponds to a bit position of the function input. To evaluate a PRF with key-matrix K on ℓ-bit input x, $\lg \ell$ synthesizer evaluations must occur. For clarity of exposition, we construct a sequence $t_1, t_2, \ldots, t_\ell$ that represents the input x. We define each t_i as $k_{x_i,i}$ and recursively define a *squeeze* function SQ to describe the PRF as follows:

$$\mathrm{SQ}\left(t_0, t_1, t_2, \cdots, t_\ell\right) = \begin{cases} S\left(\mathrm{SQ}\left(t_0, \cdots, t_{\frac{\ell}{2}-1}\right), \mathrm{SQ}\left(t_{\frac{\ell}{2}}, \cdots, t_\ell\right)\right) & \text{if } \ell > 2, \\ S(t_0, t_1) & \text{Otherwise.} \end{cases}$$

The simple case when $n = 8$ is show in figure 1.

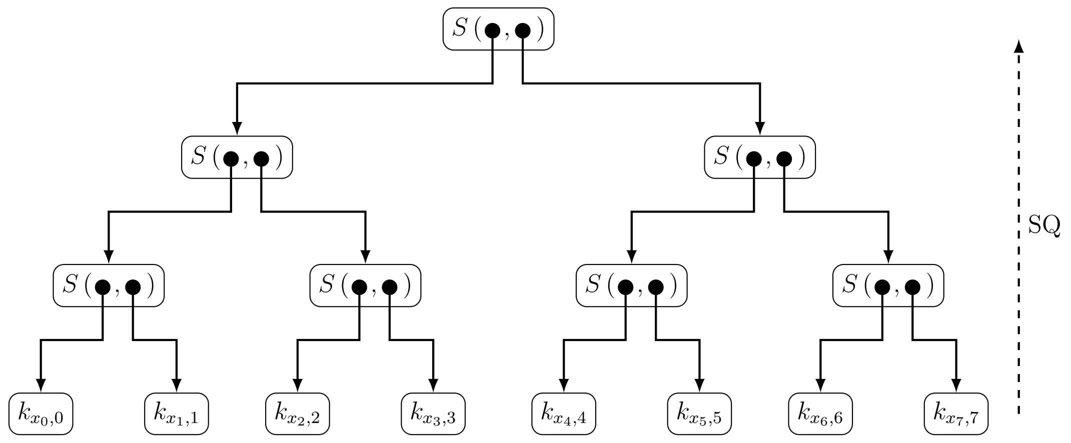

Figure 1: The evaluation of the synthesizer based PRF on input $x_0x_1x_2x_3x_4x_5x_6x_7$. Each of the levels of the tree are the result of applying the squeeze operation to the level directly below.

We note that an ℓ-dimensional pseudorandom synthesizer may be used to directly construct the Naor-Reingold PRF. The construction based on the squeeze function uses a 2-dimensional pseudorandom synthesizer from which an ℓ-dimensional one can be constructed.

A pseudorandom synthesizer can be used to combine random values in the direct-line approach, thus allowing us to realize a bit-fixed constrained PRF from one-way functions (via weak PRFs). In fact the direct-line approach applied to an n-dimensional synthesizer results in the [NR95] PRF.

3 Bit-fixing from Pseudorandom Synthesizers

Given any pseudorandom synthesizer, one can build a bit-fixing CPRF that constrains privately. The core idea is to use the PRF construction from pseudorandom synthesizers together with the direct line approach. Given the key K which is a $2 \times \ell$ matrix of values sampled uniformly from $\{0,1\}^n$, a key K_{φ_p} for a bit-fixing predicate φ_p is defined as a $2 \times \ell$

matrix where,

$$k_{\varphi_p,i,j} = \begin{cases} k_{\varphi_p,i,j} = k_{i,j} & \text{if } p_j = * \text{ or } p_j = i, \\ r_{i,j} \xleftarrow{\$} \{0,1\}^n & \text{otherwise.} \end{cases}$$

Formally our system can be realized as:

Construction 3.1. Let $S : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom synthesizer and ℓ be a power of two. We can define a CPRF $\Pi = (\mathsf{F.Gen}, \mathsf{F.Constrain}, \mathsf{F.Eval}, \mathsf{F.ConstrainedEval})$ from domain $\{0,1\}^\ell$ to range $\{0,1\}^n$ as:

- $\mathsf{F.Gen}\left(1^\lambda\right)$ return the master key $\mathsf{msk} = \left(\mathsf{K}, \hat{\mathsf{K}}\right)$ where,

  $$\mathsf{K} = \begin{pmatrix} k_{0,0}, k_{0,1}, \ldots k_{0,\ell-1} \\ k_{1,0}, k_{1,1}, \ldots k_{1,\ell-1} \end{pmatrix}, \quad k_{i,j} \xleftarrow{\$} \{0,1\}^n;$$

  and

  $$\hat{\mathsf{K}} = \begin{pmatrix} \hat{k}_{0,0}, \hat{k}_{0,1}, \ldots \hat{k}_{0,\ell-1} \\ \hat{k}_{1,0}, \hat{k}_{1,1}, \ldots \hat{k}_{1,\ell-1} \end{pmatrix}, \quad \hat{k}_{i,j} \xleftarrow{\$} \{0,1\}^n.$$

- $\mathsf{F.Constrain}\,(\mathsf{msk}, \varphi_v)$ given the master key and the bit-fixing pattern φ_v we return the key matrix

  $$\mathsf{K}_{\varphi_v} = \begin{pmatrix} k_{\varphi_v,0,0}, k_{\varphi_v,0,1}, \ldots k_{\varphi_v,0,\ell-1} \\ k_{\varphi_v,1,0}, k_{\varphi_v,1,1}, \ldots k_{\varphi_v,1,\ell-1} \end{pmatrix},$$

  where

  $$k_{\varphi_v,i,j} = \begin{cases} k_{i,j} & \text{if } v_j = * \text{ or } v_j = i, \\ \hat{k}_{i,j} & \text{otherwise.} \end{cases}$$

- $\mathsf{F.Eval}\,(\mathsf{msk}, x)$ we return the value of the recursive function $\mathrm{SQ}\left(k_{x_0,0}k_{x_1,1}\cdots k_{x_{\ell-1},\ell-1}\right)$
- $\mathsf{F.ConstrainedEval}\,(\mathsf{K}_{\varphi_v}, x)$ return the value of $\mathsf{Eval}\,(\mathsf{K}_{\varphi_v}, x)$.

In the above construction, an ℓ-way pseudorandom synthesizer can be directly substituted thus simplifying the Eval function so that only one synthesizer call is made, instead of the $\lg \ell$ needed for the two dimensional synthesizer.

Related Keys. Observe that the $\mathsf{F.CosntrainedEval}$ algorithm is nothing more than a call to $\mathsf{F.Eval}$ of the underlying PRF. In fact, we are in effect invoking the PRF with a different master key. We point out this property, as it is some what unique in the literature; the only other constructions, which we are aware of, that have this property are [DKN$^+$20] when $Q = 1$ and [PS20]. In essence, our construction allows the party that possess the master key k for the PRF to efficiently find a related key $\hat{k}$ that indexes a different PRF in the family with the property that $f_k(x) = f_{\hat{k}}(x)$ for all $x \in \mathcal{X}$ such that $\varphi(x) = 1$. Because of this, the constrained key is non-expanding (unlike GGM based CPRFs, for example).

Correctness. The intuition behind the correctness is that for all elements x where the predicate is satisfied, the key posses the same inputs to the synthesizer as the master key. Intuitively we can view the constrained key as the selection of a new PRF from the family. This gives some indication as to why this construction is pseudorandom at values of x that do not satisfy the bit-fixing predicate.

Theorem 3.1. *Construction 3.1 is correct.*

Proof. The $\mathsf{F.Constrain}$ algorithm guarantees that the elements of the constrained key K_{φ_p} will agree with the elements of master key $\left(\mathsf{K}, \hat{\mathsf{K}}\right)$ when the corresponding bit matches pattern p. For every x where $\varphi_p(x) = 1$, we know that the necessary entries of K_{φ_p} are present. Therefore when evaluation is performed the synthesizer calls will be provided with the same entries. Thus the evaluation with K will agree with the evaluation with K_{φ_p}. When $\varphi_p(x) = 0$ we have that the key K_{φ_p} will not result in $\mathsf{F.Eval}(\mathsf{K}, x) = \mathsf{F.ConstrainedEval}\big(\mathsf{K}_{\varphi_p}, x\big)$ as at least one of the bits of x_i of x disagrees with a corresponding p_i in bit pattern p. Thus one of the synthesizer values will be wrong resulting in a different output. □

Security. The security argument is similar to the hybrid argument used in [NR95]. The main difference is that our hybrids remain consistent with any values, or partial values, the adversary can compute themselves.

Theorem 3.2. *Construction 3.1 is pseudorandom at constrained points (CPRF Secure).*

Proof. We prove our result using a standard hybrid argument. We define $\lg n$ hybrids such that hybrid i behaves as the CPRF-secure game except for the handling of evaluation oracle queries. The evaluation oracle responds honestly for any query x that satisfies the constrained key's predicate. If the x does not satisfy the predicate, we replace all nodes at depth i (except those which the adversary can compute themselves) with random values. We then continue the labeling of the tree as normal. Note that H_0 is the CPRF-secure game when $b = 0$ and $H_{\lg n}$ is the CPRF-secure game when $b = 1$.

Assume by way of contradiction that $H_0 \not\approx_c H_{\lg n}$. Therefore, there must exist some distinguisher D that can distinguish H_0 from $H_{\lg n}$ with greater than negligible probability. Let ϵ be the distinguishing probability of D. By a the standard hybrid argument, there must exist some pair of neighboring distributions that can be distiguished with probability at least $\frac{\epsilon}{\lg n}$. If D can distinguish these distributions then we can use D to construct a distinguisher D' that can distinguish between the matrix of values output by a pseudorandom synthesizer and the a truly random matrix of values drawn uniformly from the same range. Assuming D makes at most some polynomial $q(n)$ queries to the evaluation oracle, the algorithm for D' takes a matrix B with dimensions $nq(n) \times nq(n)$:

$D'(B)$:

1. Sample j uniformly at random from the range $[0, \lg n)$
2. Run D to obtain the bit-fixed predicate φ.
3. Run $\mathsf{F.Gen}(1^\lambda)$ to obtain msk.
4. Run $\mathsf{F.Constrain}(\mathsf{msk}, \varphi)$ to obtain the constrained key K_φ which is given to D.
5. Extract a series of submatrices $B^{(i)}$ from B for $i \in [1, 2^j]$ where $B^{(i)} = \left(b^{(i)}_{u,v}\right)^{q(n)}_{u,v=1}$ and $b^{(i)}_{u,v} = b_u + ((i-1)\,q(n) + 1), v + ((i-1)\,q(n) + 1)$
6. When D issues an evaluation oracle query for $x \in \{0,1\}^\ell$, respond as follows:
   - If $\varphi(x) = 1$, respond with the value of $\mathsf{F.Eval}(\mathsf{msk}, x)$.
   - If $\varphi(x) = 0$, label nodes at depth j as follows:
     - if the two children x and y of the node i can be computed by D given K_φ, then label node i with $S(x, y)$.
     - if at least one of the children of node i can not be computed by D given K_φ, we label node i, that corresponds to a specific substring $s = s_0 s_1$, with the value $b^{(i)}_{u,v}$ where u is the is the row we associate with string s_0 and and v is the column we associate with string s_1. If no such association has been fixed, select one. Finally, complete the labeling of the tree by applying the synthesizer and output the label associated with the root of the tree.

Observe if B is the output of a synthesizer then D' is simulating H_j if B was generated by a pseudorandom synthesizer, D' is simulating H_{j-1}. Since, S is a pseudorandom synthesizer D' can't distinguish with better than $\mathsf{negl}(n)$ probability. Therefore, D succeeds with probability at most $\frac{1}{\lg n}\mathsf{negl}(n)$. □

One-Key Privacy. Our construction is one-key private. Intuitively given a key one can not determine the predicate used to generate that key from any other key.

Theorem 3.3. *Construction 3.1 is one-key private (CPRF predicate private).*

Proof. The constrained key for two predicates φ_0 and φ_1 will have the same entries for all x where $\varphi_0(x) = \varphi_1(x)$. Since the adversary can not query for $x \in Q$ that do not satisfy both predicates, the random values in the constrained key matrices will be indistinguishable from one another. □

4 Conclusion

We have shown that from the synthesizer based PRF construction of [NR95], a one-key private single-key selective-secure CPRF for the bit-fixing family of predicates can be constructed. Thus we resolve an open question on the richness of CPRFs based on black-box style PRF constructions. Lastly, we noted a interesting feature present in our CPRF where $\mathsf{F.ConstrainedEval}(\cdot)$ is equivalent to calling $\mathsf{F.Eval}(\cdot)$ using the constrained key, implying the size of the constrained keys and master keys are the same. We leave as an open question what other CPRFs can be constructed that have this property.

References

[AMN+18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained prfs for NC^1 in traditional groups. In *Annual International Cryptology Conference*, pages 543–574. Springer, 2018.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography–PKC 2014*, pages 501–519. Springer, 2014.

[BKM17] Dan Boneh, Sam Kim, and Hart Montgomery. Private puncturable prfs from standard lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–445. Springer, 2017.

[BLW17] Dan Boneh, Kevin Lewi, and David J Wu. Constraining pseudorandom functions privately. In *IACR International Workshop on Public Key Cryptography*, pages 494–524. Springer, 2017.

[BPS19] Florian Bourse, David Pointcheval, and Olivier Sanders. Divisible e-cash from constrained pseudo-random functions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 679–708. Springer, 2019.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from lwe. In *Theory of Cryptography Conference*, pages 264–302. Springer, 2017.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions. In *Theory of Cryptography Conference*, pages 1–30. Springer, 2015.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013*, pages 280–300. Springer, 2013.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for NC^1 from lwe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 446–476. Springer, 2017.

[DKN$^+$20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In *Annual International Cryptology Conference*, pages 559–589. Springer, 2020.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[HKKW19] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In *International Conference on Financial Cryptography and Data Security*, pages 357–376. Springer, 2019.

[Kis19] Zachary A Kissel. Key regression from constrained pseudorandom functions. *Information Processing Letters*, 2019.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.

[MS95] Silvio Micali and Ray Sidney. A simple method for generating and sharing pseudo-random functions, with applications to clipper-like key escrow systems. In *Annual International Cryptology Conference*, pages 185–196. Springer, 1995.

[NR95] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 170–181. IEEE, 1995.

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

[PS20] Chris Peikert and Sina Shiehian. Constraining and watermarking prfs from milder assumptions. In *IACR International Conference on Public-Key Cryptography*, pages 431–461. Springer, 2020.

[SW21] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. *SIAM Journal on Computing*, 50(3):857–908, 2021.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-cnf from lwe. In *Annual International Cryptology Conference*, pages 62–85. Springer, 2019.