Constrained Pseudorandom Functions and Their Applications*

Dan Boneh[1] and Brent Waters[2]

[1] Stanford University
dabo@cs.stanford.edu
[2] U.T. Austin
bwaters@cs.utexas.edu

Abstract. We put forward a new notion of pseudorandom functions (PRFs) we call constrained PRFs. In a standard PRF there is a master key k that enables one to evaluate the function at all points in the domain of the function. In a constrained PRF it is possible to derive constrained keys k_S from the master key k. A constrained key k_S enables the evaluation of the PRF at a certain subset S of the domain and nowhere else. We present a formal framework for this concept and show that constrained PRFs can be used to construct powerful primitives such as identity-based key exchange and a broadcast encryption system with optimal ciphertext size. We then construct constrained PRFs for several natural set systems needed for these applications. We conclude with several open problems relating to this new concept.

1 Introduction

Pseudorandom functions(PRF) [20] are a fundamental concept in modern cryptography. A PRF is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm: on input $(k, x) \in \mathcal{K} \times \mathcal{X}$ the algorithm outputs $F(k, x) \in \mathcal{Y}$. Note that given the key $k \in \mathcal{K}$, the function $F(k, \cdot)$ can be efficiently evaluated at *all* points $x \in \mathcal{X}$.

In this paper we put forward a new notion of PRFs we call *constrained PRFs*. Consider a PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and let $k_0 \in \mathcal{K}$ be some key for F. In a constrained PRF one can derive constrained keys k_S from the master PRF key k_0. Each constrained key k_S corresponds to some subset $S \subseteq X$ and enables one to evaluate the function $F(k_0, x)$ for $x \in S$, but at no other points in the domain $\mathcal{X}$. A constrained PRF is secure if given several constrained keys for sets $S_1, \ldots, S_q$ of the adversary's choice, the adversary cannot distinguish the PRF from random for points x outside these sets, namely for $x \notin \cup_{i=1}^{q} S_i$. We give precise definitions in Section 3.

While constrained PRFs are a natural extension of the standard concept of PRFs, they have surprisingly powerful applications beyond what is possible with standard PRFs. We list a few examples here and present more applications in Section 6:

* The full version is available as Cryptology ePrint Archive, Report 2013/352.

K. Sako and P. Sarkar (Eds.) ASIACRYPT 2013, Part II, LNCS 8270, pp. 280–300, 2013.

- **Left-Right PRFs:** Let $F : \mathcal{K} \times \mathcal{X}^2 \to \mathcal{Y}$ be a secure PRF. Its domain is $\mathcal{X} \times \mathcal{X}$. Now, suppose that for every $w \in \mathcal{X}$ there are two constrained keys $k_{w,\text{left}}$ and $k_{w,\text{right}}$. The key $k_{w,\text{left}}$ enables the evaluation of $F(k_0, \cdot)$ at the subset of points $\{(w, y) : y \in \mathcal{X}\}$ (i.e. at all points where the left side is w). The key $k_{w,\text{right}}$ enables the evaluation of $F(k_0, \cdot)$ at the subset of points $\{(x, w) : x \in \mathcal{X}\}$ (i.e. at all points where the right side is w). We show that such a constrained PRF can be used to construct an identity-based non-interactive key exchange (ID-NIKE) system [31,14,27,16].
- **Bit-Fixing PRFs:** Let $\mathcal{X} = \{0,1\}^n$ be the domain of the PRF. For a vector $v \in \{0, 1, ?\}^n$ let $S_v \subseteq \mathcal{X}$ be the set of n-bit strings that match v at all the coordinates where v is not '?'. We say that S_v is bit-fixed to v. For example, the set containing all n-bit strings starting with 00 and ending in 11 is bit-fixed to $v = 00?\ldots?11$.
  Now, suppose that for every bit-fixed subset S of $\{0,1\}^n$ there is a constrained key k_S that enables the evaluation of $F(k_0, x)$ at $x \in S$ and nowhere else. We show that such a constrained PRF can be used to construct an *optimal* secret-key[1] broadcast encryption system [15]. In particular, the length of the private key and the broadcast ciphertext are all *independent* of the number of users. We compare these constructions to existing broadcast systems in Section 6.1.
- **Circuit PRFs:** Let $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ be a secure PRF. Suppose that for every polynomial size circuit C there is a constrained key k_C that enables the evaluation of $F(k_0, x)$ at all points $x \in \{0,1\}^n$ such that $C(x) = 1$. We show that such a constrained PRF gives rise to a non-interactive policy-based key exchange mechanism: a group of users identified by a complex policy (encoded as a circuit) can non-interactively setup a secret group key that they can then use for secure communications among group members. A related concept was studied by Gorantla et al. [21], but the schemes presented are interactive, analyzed in the generic group model, and only apply to policies represented as polynomial size formulas.

In the coming sections we present constructions for all the constrained PRFs discussed above as well as several others. Some of our constructions use bilinear maps while others require κ-linear maps [7,17,11] for $\kappa > 2$. It would be quite interesting and useful to develop constructions for these constrained PRFs from other assumptions such as Learning With Errors (LWE) [28]. This will give new key exchange and broadcast encryption systems from the LWE problem.

In defining security for a constrained PRF in Section 3 we allow the adversary to adaptively request constrained keys of his choice. The adversary's goal is to distinguish the PRF from a random function at input points where he cannot compute the PRF using the constrained keys at his disposal. The definition of security allows the adversary to *adaptively* choose the challenge point at which he tries to distinguish the PRF from random. However, to prove security of our constructions we require that the attacker commit to the challenge point ahead

[1] Secret-key broadcast encryption refers to the fact that the broadcaster's key is known only to the broadcaster.

of time thereby only proving a weaker notion of security called selective security. A standard argument called *complexity leveraging* (see e.g. [4, Sec. 7.1]) shows that selective security implies adaptive security via a non-polynomial time reduction. Therefore, to obtain adaptive security we must increase the parameters of our schemes so that security is maintained under the complexity leveraging reduction. A fascinating open problem is to construct standard model constrained PRFs that are adaptively secure under a polynomial time reduction.

Related work. Concurrently with this paper, similar notions to constrained PRFs were recently proposed by Kiayias et al. [24] where they were called delegatable PRFs and Boyle et al. [9] where they were called functional PRFs. Both papers give constructions for prefix constraints discussed in Section 3.3. A related concept applied to digital signatures was explored by Bellare and Fuchsbauer [1] where it was called policy-based signatures and by Boyle et al. [9] where it was called functional signatures.

2 Preliminaries: Bilinear and κ-Linear Maps

Recently, Garg, Gentry, and Halevi [17] proposed candidate constructions for leveled multilinear forms. Building on their work Coron, Lepoint, and Tibouchi [11] gave a second candidate. We will present some of our constructions using the abstraction of leveled multilinear groups.

The candidate constructions of [17,11] implement an abstraction called graded encodings which is similar, but slightly different from multilinear groups. In the full version [8] we show how to map our constructions to the language of graded encodings.

Leveled multilinear groups. We assume the existence of a group generator $\mathcal{G}$, which takes as input a security parameter 1^λ and a positive integer κ to indicate the number of levels. $\mathcal{G}(1^\lambda, \kappa)$ outputs a sequence of groups $\boldsymbol{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ each of large prime order $p > 2^\lambda$. In addition, we let g_i be a canonical generator of $\mathbb{G}_i$ that is known from the group's description. We let $g = g_1$.

We assume the existence of a set of bilinear maps $\{e_{i,j} : G_i \times G_j \to G_{i+j} \mid i, j \geq 1;\ i + j \leq \kappa\}$. The map $e_{i,j}$ satisfies the following relation:

$$e_{i,j}\left(g_i^a, g_j^b\right) = g_{i+j}^{ab} \ : \ \forall a, b \in \mathbb{Z}_p$$

We observe that one consequence of this is that $e_{i,j}(g_i, g_j) = g_{i+j}$ for each valid i, j. When the context is obvious, we will sometimes drop the subscripts i, j, For example, we may simply write:

$$e\left(g_i^a, g_j^b\right) = g_{i+j}^{ab}.$$

We define the κ-Multilinear Decisional Diffie-Hellman (κ-MDDH) assumption as follows:

Assumption 1 (κ-Multilinear Decisional Diffie-Hellman: κ-MDDH) *The κ-Multilinear Decisional Diffie-Hellman (κ-MDDH) problem states the following: A challenger runs $\mathcal{G}(1^\lambda, \kappa)$ to generate groups and generators of order p. Then it picks random $c_1, \ldots, c_{\kappa+1} \in \mathbb{Z}_p$.*

The assumption then states that given $g = g_1, g^{c_1}, \ldots, g^{c_{\kappa+1}}$ it is hard to distinguish the element $T = g_\kappa^{\prod_{j\in[1,\kappa+1]} c_j} \in \mathbb{G}_\kappa$ from a random group element in $\mathbb{G}_\kappa$, with better than negligible advantage in the security parameter λ.

3 Constrained Pseudorandom Functions

We now give a precise definition of constrained Pseudorandom Functions. We begin with the syntax of the constrained PRF primitive and then define the security requirement.

3.1 The Constrained PRF Framework

Recall that a pseudorandom function (PRF) [20] is defined over a key space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$ (and these sets may be parameterized by the security parameter λ). The PRF itself is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm: on input $(k, x) \in \mathcal{K} \times \mathcal{X}$ the algorithm outputs $F(k, x) \in \mathcal{Y}$. A PRF can include a setup algorithm $F.\mathsf{setup}(1^\lambda)$ that takes a security parameter λ as input and outputs a random secret key $k \in \mathcal{K}$.

A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is said to be *constrained* with respect to a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if there is an additional key space $\mathcal{K}_c$ and two additional algorithms $F.\mathsf{constrain}$ and $F.\mathsf{eval}$ as follows:

- $F.\mathsf{constrain}(k, S)$ is a randomized polynomial-time algorithm that takes as input a PRF key $k \in \mathcal{K}$ and the description of a set $S \in \mathcal{S}$ (so that $S \subseteq \mathcal{X}$). The algorithm outputs a constrained key $k_S \in \mathcal{K}_c$. This key k_S enables the evaluation of $F(k, x)$ for all $x \in S$ and no other x.
- $F.\mathsf{eval}(k_S, x)$ is a deterministic polynomial-time algorithm (in λ) that takes as input a constrained key $k_s \in \mathcal{K}_c$ and an $x \in \mathcal{X}$. If k_S is the output of $F.\mathsf{constrain}(k, S)$ for some PRF key $k \in \mathcal{K}$ then $F.\mathsf{eval}(k_S, x)$ outputs

$$F.\mathsf{eval}(k_S, x) = \begin{cases} F(k, x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

where $\perp \notin \mathcal{Y}$. As shorthand we will occasionally write $F(k_S, x)$ for $F.\mathsf{eval}(k_S, x)$.

Note that while in general deciding if $x \in S$ may not be a poly-time problem, our formulation of $F.\mathsf{eval}$ effectively avoids this complication by requiring that all $S \in \mathcal{S}$ are poly-time decidable by the algorithm $F.\mathsf{eval}(k_S, \cdot)$. This poly-time

algorithm outputs non-$\perp$ when $x \in S$ and $\perp$ otherwise thereby deciding S in polynomial time.

Occasionally it will be convenient to treat the set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ as a family of predicates $\mathrm{PP} = \{p : \mathcal{X} \to \{0,1\}\}$. For a predicate $p \in \mathrm{PP}$ we have $F.\mathsf{eval}(k_p, x) = F(k, x)$ whenever $p(x) = 1$ and $\perp$ otherwise. In this case we say that the PRF F is constrained with respect to the family of predicates PP.

The trivial constrained PRF. All PRFs $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ are constrained with respect to the set system $\mathcal{S}$ consisting of all singleton sets: $\mathcal{S} = \{\{x\} : x \in \mathcal{X}\}$. To see why, fix some PRF key $k \in \mathcal{K}$. Then the constrained key $k_{\{x\}}$ for the singleton set $\{x\}$ is simply $k_{\{x\}} = F(k, x)$. Given this key $k_{\{x\}}$, clearly anyone can evaluate $F(k, x)$ at the point x. This shows that we may assume without loss of generality that set systems $\mathcal{S}$ used to define a constrained PRF contain all singleton sets. More generally, we may also assume that $\mathcal{S}$ contains all *polynomial size* sets (polynomial in the security parameter λ). The constrained key k_S for a polynomial size set $S \subseteq \mathcal{X}$ is simply the set of values $F(k, x)$ for all $x \in S$. This construction fails for super-polynomial size sets since the constrained key k_S for such sets is too large.

3.2 Security of Constrained Pseudorandom Functions

Next, we define the security properties of constrained PRFs. The definition captures the property that given several constrained keys as well as several function values at points of the attacker's choosing, the function looks random at all points that the attacker cannot compute himself.

Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a constrained PRF with respect to a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$. We define constrained security using the following two experiments denoted EXP(0) and EXP(1) with an adversary $\mathcal{A}$. For $b = 0, 1$ experiment EXP(b) proceeds as follows:

> First, a random key $k \in \mathcal{K}$ is selected and two helper sets $C, V \subseteq \mathcal{X}$ are initialized to $\emptyset$. The set $V \subseteq \mathcal{X}$ will keep track of all the points at which the adversary can evaluate $F(k, \cdot)$. The set $C \subseteq \mathcal{X}$ will keep track of the points where the adversary has been challenged. The sets C and V will ensure that the adversary cannot trivially decide whether challenge values are random or pseudorandom. In particular, the experiments maintain the invariant that $C \cap V = \emptyset$.
>
> The adversary $\mathcal{A}$ is then presented with three oracles as follows:
>
> - $F.\mathsf{eval}$: given $x \in \mathcal{X}$ from $\mathcal{A}$, if $x \notin C$ the oracle returns $F(k, x)$ and otherwise returns $\perp$. The set V is updated as $V \leftarrow V \cup \{x\}$.
> - $F.\mathsf{constrain}$: given a set $S \in \mathcal{S}$ from $\mathcal{A}$, if $S \cap C = \emptyset$ the oracle returns a key $F.\mathsf{constrain}(k, S)$ and otherwise returns $\perp$. The set V is updated as $V \leftarrow V \cup S$.
> - $\mathsf{Challenge}$: given $x \in \mathcal{X}$ from $\mathcal{A}$ where $x \notin V$, if $b = 0$ the adversary is given $F(k, x)$; otherwise the adversary is given a random (consistent) $y \in \mathcal{Y}$. The set C is updated as $C \leftarrow C \cup \{x\}$.

Once the adversary $\mathcal{A}$ is done interrogating the oracles it outputs $b' \in \{0,1\}$.

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\mathrm{EXP}(b)$. We define the adversary's advantage as $\mathrm{AdvPRF}_{\mathcal{A},F}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 1. *The PRF F is a secure constrained PRF with respect to $\mathcal{S}$ if for all probabilistic polynomial time adversaries $\mathcal{A}$ the function $AdvPRF_{\mathcal{A},F}(\lambda)$ is negligible.*

When constructing constrained functions it will be more convenient to work with a definition that slightly restricts the adversary's power, but is equivalent to Definition 1. In particular, we only allow the adversary to issue a *single* challenge query (but multiple queries to the other two oracles). A standard hybrid argument shows that a PRF secure under this restricted definition is also secure under Definition 1.

3.3 Example Predicate Families

Next we introduce some notation to capture the predicate families described in the introduction.

Bit-Fixing Predicates. Let $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ be a PRF. We wish to support constrained keys $k_{\mathbf{v}}$ that enable the evaluation of $F(k,x)$ at all points x that match a particular bit pattern. To do so define for a vector $\mathbf{v} \in \{0,1,?\}^n$ the predicate $p^{(\mathrm{BF})}_{\mathbf{v}} : \{0,1\}^n \to \{0,1\}$ as

$$p^{(\mathrm{BF})}_{\mathbf{v}}(x) = 1 \quad \Longleftrightarrow \quad (\mathbf{v}_i = x_i \text{ or } \mathbf{v}_i = ?) \text{ for all } i = 1, \ldots, n \ .$$

We say that $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ supports bit fixing if it is constrained with respect to the set of predicates

$$\mathcal{P}_{\mathrm{BF}} = \left\{ p^{(\mathrm{BF})}_{\mathbf{v}} \ : \ \mathbf{v} \in \{0,1,?\} \right\}$$

Prefix Predicates. Prefix predicates are a special case of bit fixing predicates in which only the prefix is fixed. More precisely, we say that $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ supports prefix fixing if it is constrained with respect to the set of predicates

$$\mathcal{P}_{\mathrm{PRE}} = \left\{ p^{(\mathrm{BF})}_{\mathbf{v}} \ : \ \mathbf{v} \in \{0,1\}^{\ell} \ ?^{n-\ell}, \ \ \ell \in [n] \right\}$$

Secure PRFs that are constrained with respect to $\mathcal{P}_{\mathrm{PRE}}$ can be constructed directly from the GGM PRF construction [20]. For a prefix $\mathbf{v} \in \{0,1\}^{\ell}$ the constrained key $k_{\mathbf{v}}$ is simply the secret key in the GGM tree computed at the internal node associated with the string $\mathbf{v}$. Clearly this key enables the evaluation of $F(k, \ \mathbf{v}\|x)$ for any $x \in \{0,1\}^{n-|\mathbf{v}|}$. A similar construction, in a very different context, was used by Fiat and Naor [15] and later by Naor, Naor, and Lotspiech [25] to construct combinatorial broadcast encryption systems. The security proof for this GGM-based prefix constrained PRF is straight forward if the adversary commits to his challenge point ahead of time (a.k.a selective security). Full security can be achieved, for example, using standard complexity leveraging by guessing the adversary's challenge point ahead of time as in [4, Sec. 7.1].

Left/Right Predicates. Let $F : \mathcal{K} \times \mathcal{X}^2 \to \mathcal{Y}$ be a PRF. For all $w \in \mathcal{X}$ we wish to support constrained keys $k_{w,\text{LEFT}}$ that enable the evaluation of $F\big(k,\ (x,y)\big)$ at all points $(w,y) \in \mathcal{X}^2$, that is, at all points in which the left side is fixed to w. In addition, we want constrained keys $k_{w,\text{RIGHT}}$ that fix the right hand side of (x,y) to w. More precisely, for an element $w \in \mathcal{X}$ define the two predicates $p_w^{(\text{L})}, p_w^{(\text{R})} : \mathcal{X}^2 \to \{0,1\}$ as

$$p_w^{(\text{L})}(x,y) = 1 \iff x = w \qquad \text{and} \qquad p_w^{(\text{R})}(x,y) = 1 \iff y = w$$

We say that F supports left/right fixing if it is constrained with respect to the set of predicates

$$\mathcal{P}_{\text{LR}} = \big\{p_w^{(\text{L})},\ p_w^{(\text{R})} \ :\ w \in \mathcal{X}\big\}$$

Constructing left/right constrained PRFs. We next show that secure PRFs that are constrained with respect to $\mathcal{P}_{\text{LR}}$ can be constructed straightforwardly in the random oracle model [3]. Constructing left/right constrained PRFs *without* random oracles is a far more challenging problem. We do so, and more, in the next section.

To construct a left/right constrained PRF in the random oracle model let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p. Let $H_1, H_2 : \mathcal{X} \to \mathbb{G}$ be two hash functions that will be modeled as random oracles. The setup algorithm will choose such a group and a random key $k \in \mathbb{Z}_p$. Define the following PRF:

$$F\big(k,\ (x,y)\big) = e\big(H_1(x), H_2(y)\big)^k \ . \tag{1}$$

For $(x^*, y^*) \in \mathcal{X}^2$ the constrained keys for the predicates $p_{x^*}^{(\text{L})}$ and $p_{y^*}^{(\text{R})}$ are

$$k_{x^*} = H_1(x^*)^k \qquad \text{and} \qquad k_{y^*} = H_2(y^*)^k$$

respectively. Clearly k_{x^*} is sufficient for evaluating $f(k,y) = F\big(k,\ (x^*,y)\big)$ and k_{y^*} is sufficient for evaluating $g(k,x) = F\big(k,\ (x,y^*)\big)$, as required. We note the structural similarities between the above construction and the Boneh-Franklin [5] IBE system and the Sakai-Ohgishi-Kasahara [31] non-interactive key exchange system.

Theorem 2. *The PRF F defined in Eq. (1) is a secure constrained PRF with respect to $\mathcal{P}_{LR}$ assuming the decision bilinear Diffie-Hellman assumption (DBDH) holds for $(\mathbb{G}, G_T, e)$ and the functions H_1, H_2 are modeled as random oracles.*

Due to space constraints the proof, which uses a standard argument, is given in the full version of the paper [8].

Circuit Predicates. Let $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ be a PRF. For a boolean circuit c on n inputs we wish to support a constrained key k_c that enable the evaluation of $F(k,x)$ at all points $x \in \mathcal{X}$ for which $c(x) = 1$.

Let $\mathcal{C}$ be the set of polynomial size circuits. We say that F supports circuit predicates if it is constrained with respect to the set of predicates

$$\mathcal{P}_{\text{circ}} = \big\{c\ :\ c \in \mathcal{C}\big\}$$

4 A Bit-Fixing Construction

We now describe our bit-fixing constrained PRF. We will present our construction in terms of three algorithms which include a setup algorithm $F.\mathsf{setup}$ in addition to $F.\mathsf{constrain}$ and $F.\mathsf{eval}$. Our construction builds on the Naor-Reingold DDH-based PRF [26].

4.1 Construction

$F.\mathsf{setup}(1^\lambda, 1^n)$:
The setup algorithm takes as input the security parameter λ and the bit length, n, of PRF inputs. The algorithm runs $\mathcal{G}(1^\lambda, \kappa = n+1)$ and outputs a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ of prime order p, with canonical generators $g_1, \ldots, g_\kappa$, where we let $g = g_1$. It then chooses random exponents $\alpha \in \mathbb{Z}_p$ and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in {\mathbb{Z}_p}^2$ and computes $D_{i,\beta} = g^{d_{i,\beta}}$ for $i \in [1,n]$ and $\beta \in \{0,1\}$. The PRF master key k consists of the group sequence $(\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ along with $\alpha, d_{i,\beta}$ and $D_{i,\beta}$ for $i \in [1,n]$ and $\beta \in \{0,1\}$.

The domain $\mathcal{X}$ is $\{0,1\}^n$ and the range of the function is $\mathbb{G}_k$.[2] Letting x_i denote the i-th bit of $x \in \{0,1\}^n$, the keyed function is defined as

$$F(k,x) = g_\kappa^{\alpha \prod_{i \in [1,n]} d_{i,x_i}} \quad \in \mathbb{G}_\kappa \ .$$

$F.\mathsf{constrain}(k, \mathbf{v})$:
The constrain algorithm takes as input the master key k and a vector $\mathbf{v} \in \{0,1,?\}^n$. (Here we use the vector $\mathbf{v}$ to represent the set for which we want to allow evaluation.) Let V be the set of indices $i \in [1,n]$ such that $\mathbf{v}_i \neq ?$. That is the the indices for which the bit is fixed to 0 or 1.

The first component of the constrained key is computed as

$$k'_{\mathbf{v}} = (g_{1+|V|})^{\alpha \prod_{i \in V} d_{i,\mathbf{v}_i}}$$

Note if V is the empty set we interpret the product to be 1. The constrained key $k_{\mathbf{v}}$ consists of $k'_{\mathbf{v}}$ along with $D_{i,\beta}$ $\forall i \notin V, \beta \in \{0,1\}$.

$F.\mathsf{eval}(k_{\mathbf{v}}, x)$:
Again let V be the set of indices $i \in [1,n]$ such that $\mathbf{v}_i \neq ?$. If $\exists i \in V$ such that $x_i \neq \mathbf{v}_i$ the algorithm aborts. If $|V| = n$ then all bits are fixed and the output of the function is $k_{\mathbf{v}}$. Otherwise, using repeated application of the pairing and $D_{i,\beta}$ $\forall i \notin V, \beta \in \{0,1\}$ the algorithm can compute the intermediate value

$$T = (g_{n-|V|})^{\prod_{i \in [1,n] \setminus V} (d_{i,x_i})} \ .$$

Finally, it computes $e(T, k'_{\mathbf{v}}) = g_\kappa^{\alpha \prod_{i \in [1,n]} d_{i,x_i}} = F(k,x)$.

[2] In practice one can use an extractor on the output to produce a bit string.

A few notes. We note that the values $D_{i,\beta} = g^{d_{i,\beta}}$ for $i \in [1,n]$ and $\beta \in \{0,1\}$ could either be computed in setup and stored or computed as needed during the $F.\mathsf{constrain}$ function. As an alternative system one might save storage by utilizing a trusted common setup and make the group description plus the $D_{i,\beta}$ values public. These values would be shared and only the α parameter would be chosen per key. Our proof though will focus solely on the base system described above.

In the full version [8] we show how to map the construction above stated using multilinear maps to the language of graded encodings for which [17,11] provide a candidate instantiation.

4.2 Proof of Security

To show that our bit-fixing construction is secure we show that for an n-bit domain, if the $\kappa = n+1$-Multilinear Decisional Diffie-Hellman assumption holds then our construction is secure for appropriate choice of the group generator security parameter.

As stated in Section 3 a standard hybrid argument allows us to prove security in a definition where the attacker is allowed a single query x^* to the challenge oracle. Our proof will use the standard complexity leveraging technique of guessing the challenge x^* technique to prove adaptive security. The guess will cause a loss of $1/2^n$ factor in the reduction. An interesting problem is to prove security with only a polynomial factors. The reduction will program all values of $D_{i,\beta}$ to be g_i^c if $x_i = \beta$ and g^{z_i} otherwise for known z_i.

Theorem 3. *If there exists a poly-time attack algorithm $\mathcal{A}$ that breaks our bit-fixing construction n-bit input with advantage $\epsilon(\lambda)$ there there exists a poly-time algorithm $\mathcal{B}$ that breaks the $\kappa = n+1$-Multilinear Decisional Diffie-Hellman assumption with advantage $\epsilon(\lambda)/2^n$.*

Proof. We show how to construct $\mathcal{B}$. The algorithm $\mathcal{B}$ first receives an $\kappa = n+1$-MDDH challenge consisting of the group sequence description $\boldsymbol{G}$ and $g = g_1, g^{c_1}, \ldots, g^{c_{\kappa+1}}$ along with T where T is either $g_k^{\prod_{j\in[1,k+1]} c_j}$ or a random group element in $\mathbb{G}_\kappa$. It then chooses a value $x^* \in \{0,1\}^n$ uniformly at random. Next, it chooses random $z_1, \ldots, z_n$ (internally) sets

$$D_{i,\beta} = \begin{cases} g^{c_i} \text{ if } x_i^* = \beta \\ g^{z_i} \text{ if } x_i^* \neq \beta \end{cases}$$

for $i \in [1,n], \beta \in \{0,1\}$. This corresponds to setting $d_{i,\beta} = c_i$ if $x_i^* = \beta$ and z_i otherwise. We observe this is distributed identically to the real scheme. In addition, it will internally view $\alpha = c_k \cdot c_{k+1}$.

Constrain Oracle We now describe how the algorithm responds to the key query oracle. Suppose a query is made for a secret key for $\mathbf{v} \in \{0,1,?\}^n$. Let V be the set of indices $i \in [1,n]$ such that $\mathbf{v}_i \neq ?$. That is the the indices for which the bit

is fixed to 0 or 1. $\mathcal{B}$ identifies an arbitrary $i \in V$ such that $\mathbf{v}_i \neq x_i^*$. If no such i exists this means that the key cannot be produced since it could be used to evaluate $F(k, x^*)$. In this case abort and output a random guess for $\delta' \in \{0,1\}$.

If the query did not cause an abort, $\mathcal{B}$ first computes $g_2^\alpha = e(g^{c_k}, g^{c_{k+1}})$. It then gathers all $D_{j,\mathbf{v}_j}$ for $j \in V/i$. It uses repeated application of the pairing with these values to compute $(g_{1+|V|})^{\alpha \prod_{j \in V/i} d_{j,\mathbf{v}_j}}$. (Recall, our previous assignments to d_j, β.) Finally, it raises this value to $d_{i,\mathbf{v}_I} = z_i$ which is known to the attacker to get. $k'_{vv} = (g_{1+|V|})^{\alpha \prod_{j \in V/i} d_{j,\mathbf{v}_j}}$. The rest of the key is simply the $D_{j,\beta}$ values for $j \notin V, \beta \in \{0,1\}$.

Evaluate Oracle To handle the evaluation oracle, we observe that the output of $F(k, x)$ for $x \in \{0,1\}$ is identical to asking a key for $k_{\mathbf{v}=x}$ (a key with no ? symbols. Therefore, queries to this oracle can be handled as secret key queries described above.

Challenge Finally, the attacker can query a challenge oracle once. If the query to this oracle is not equal to x^* then $\mathcal{B}$ randomly guesses $\delta' \in \{0,1\}$. Otherwise, it outputs T as a response to the oracle query.

The attack algorithm will eventually output a guess b'. If $\mathcal{B}$ has not aborted, it will simply output $\delta' = b'$.

We now analyze the probability that $\mathcal{B}$'s guess $\delta' = \delta$, where δ indicates if T was an MDDH tuple. We have

$$\begin{aligned}
\Pr[\delta' = \delta] &= \Pr[\delta' = \delta | \mathsf{abort}] \cdot \Pr[\mathsf{abort}] + \Pr[\delta' = \delta | \overline{\mathsf{abort}}] \cdot \Pr[\overline{\mathsf{abort}}] \\
&= \frac{1}{2}(1 - 2^{-n}) + \Pr[\delta' = \delta | \overline{\mathsf{abort}}] \cdot (2^{-n}) \\
&= \frac{1}{2}(1 - 2^{-n}) + (\frac{1}{2} + \epsilon(\lambda)) \cdot (2^{-n}) \\
&= \frac{1}{2} + \epsilon(\lambda) \cdot (2^{-n})
\end{aligned}$$

The set of equations shows that the advantage of $\mathcal{B}$ is $\epsilon(\lambda)2^{-n}$. The second equation is derived since the probability of $\mathcal{B}$ not aborting is 2^{-n}. The third equation comes from the fact that the probability of the attacker winning given a conditioned on not aborting is the same as the original probability of the attacker winning. The reason is that the attacker's success is independent of whether $\mathcal{B}$ guessed x^*. This concludes the proof.

5 Constrained PRFs for Circuit Predicates

Next we build constrained PRFs where the accepting set for a key can be described by a polynomial size circuit. Our construction utilizes the structure used in a recent Attribute-Based Encryption scheme due to Garg, Gentry, Halevi, Sahai, and Waters [18].

We present our circuit construction for constrained PRFs in terms of three algorithms which include a setup algorithm $F.\mathsf{setup}$ in addition to $F.\mathsf{constrain}$

and $F.\mathsf{eval}$. The setup algorithm will take an additional input ℓ which is the maximum depth of circuits allowed. For simplicity we assume all circuits are depth ℓ and are leveled. We use the same notation for circuits as in [18]. We include the notation in Appendix A for completeness. In addition, like [18] we also build our construction for monotone circuits (limiting ourselves to AND and OR gates); however, we make the standard observation that by pushing NOT gates to the input wires using De Morgan's law we obtain the same result for general circuits.

5.1 Construction

$F.\mathsf{setup}(1^\lambda, 1^n, 1^\ell)$:
The setup algorithm takes as input the security parameter λ and the bit length, n, of inputs to the PRF and ℓ the maximum depth of the circuit. The algorithm runs $\mathcal{G}(1^\lambda, \kappa = n + \ell)$ and outputs a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ of prime order p, with canonical generators $g_1, \ldots, g_\kappa$, where we let $g = g_1$. It then chooses random exponents $\alpha \in \mathbb{Z}_p$ and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in {\mathbb{Z}_p}^2$ and computes $D_{i,\beta} = g^{d_{i,\beta}}$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$. The key k consists group sequence $(\mathbb{G}_1, \ldots, \mathbb{G}_k)$ along with $\alpha, d_{i,\beta}$ and $D_{i,\beta}$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$.

The domain $\mathcal{X}$ is $\{0, 1\}^n$ and the range of the function is $\mathbb{G}_k$. Letting x_i denote the i-th bit of $x \in \{0, 1\}^n$, the keyed function is defined as

$$F(k, x) = g_\kappa^{\alpha \prod_{i \in [1,n]} d_{i,x_i}} \quad \in \mathbb{G}_\kappa \quad .$$

$F.\mathsf{constrain}\big(k,\ f = (n, q, A, B, \mathtt{GateType})\big)$:
The constrain algorithm takes as input the key and a circuit description f. The circuit has $n + q$ wires with n input wires, q gates and the wire $n + q$ designated as the output wire.

To generate a constrained key k_f the key generation algorithm chooses random $r_1, \ldots, r_{n+q-1} \in \mathbb{Z}_p$, where we think of the random value r_w as being associated with wire w. It sets $r_{n+q} = \alpha$. The first part of the constrained key is given out as simply all $D_{i,\beta}$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$.

Next, the algorithm generates key components for every wire w. The structure of the key components depends upon if w is an input wire, an OR gate, or an AND gate. We describe how it generates components for each case.

- *Input wire*
  By our convention if $w \in [1, n]$ then it corresponds to the w-th input. The key component is:
  $$K_w = g_2^{r_w d_{w,1}}$$
- *OR gate*
  Suppose that wire $w \in$ Gates and that $\mathtt{GateType}(w) = \mathrm{OR}$. In addition, let $j = \mathtt{depth}(w)$ be the depth of wire w. The algorithm will choose random $a_w, b_w \in \mathbb{Z}_p$. Then the algorithm creates key components:
  $$K_{w,1} = g^{a_w},\ K_{w,2} = g^{b_w},\ K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)}},\ K_{w,4} = g_j^{r_w - b_w \cdot r_{B(w)}}$$

– *AND gate*
Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{AND}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. The algorithm will choose random $a_w, b_w \in \mathbb{Z}_p$.

$$K_{w,1} = g^{a_w},\ K_{w,2} = g^{b_w},\ K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}$$

The constrained key k_f consists of all these $n + q$ key components along with $\{D_{i,\beta}\}$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$.

$F.\mathsf{eval}(k_f, x)$:
The evaluation algorithm takes as input k_f for circuit $f = (n, q, A, B, \texttt{GateType})$ and an input x. The algorithm first checks that $f(x) = 1$; it not it aborts.

The goal of the algorithm is to compute $F(k, x) = (g_{\kappa=n+\ell})^{\alpha \prod_{i \in [1,n]} d_{i,x_i}}$. We will evaluate the circuit from the bottom up. Consider wire w at depth j; if $f_w(x) = 1$ then, our algorithm will compute $E_w = (g_{j+n})^{r_w \prod_i d_{i,x_i}}$. (If $f_w(x) = 0$ nothing needs to be computed for that wire.) Our decryption algorithm proceeds iteratively starting with computing E_1 and proceeds in order to finally compute E_{n+q}. Computing these values in order ensures that the computation on a depth $j - 1$ wire (that evaluates to 1) will be defined before computing for a depth j wire. Since $r_{n+q} = \alpha$, $E_{n+q} = F(k, x)$.

We show how to compute E_w for all w where $f_w(x) = 1$, again breaking the cases according to whether the wire is an input, AND or OR gate.

– *Input wire*
By our convention if $w \in [1, n]$ then it corresponds to the w-th input. Suppose that $x_w = f_w(x) = 1$. The algorithm computes $E_w = g_{n+1}^{r_w \prod_i d_{i,x_i}}$. Using the pairing operation successively it can compute $g_{n-1}^{\prod_{i \neq w} d_{i,x_i}}$ from the values $D_{x_i,\beta}$ for $i \in [1, n] \neq w$. It then computes

$$E_w = e(K_w, g_{n-1}^{\prod_{i \neq w} d_{i,x_i}}) = e(g_2^{r_w d_{w,1}}, g_{n-1}^{\prod_{i \neq w} d_{i,x_i}}) = g_{n+1}^{r_w \prod_i d_{i,x_i}}$$

– *OR gate*
Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{OR}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. For exposition we define $D(x) = g_n^{\prod_i d_{i,x_i}}$. This is computable via the pairing operation from $D_{x_i,\beta}$ for $i \in [1, n]$.
The computation is performed if $f_w(x) = 1$. If $f_{A(w)}(x) = 1$ (the first input evaluated to 1) then we compute:

$$E_w = e(E_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, D(x)) =$$
$$= e((g_{j+n-1})^{r_{A(w)} \prod_i d_{i,x_i}}, g^{a_w}) \cdot e(g_j^{r_w - a_w \cdot r_{A(w)}}, g_n^{\prod_i d_{i,x_i}}) = (g_{j+n})^{r_w g_n^{\prod_i d_{i,x_i}}}$$

Otherwise, if $f_{A(w)}(x) = 0$, but $f_{B(w)}(x) = 1$, then we compute:

$$E_w = e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, D(x)) =$$
$$= e((g_{j+n-1})^{r_{B(w)} \prod_i d_{i,x_i}}, g^{b_w}) \cdot e(g_j^{r_w - b_w \cdot r_{B(w)}}, g_n^{\prod_i d_{i,x_i}}) = (g_{j+n})^{r_w g_n^{\prod_i d_{i,x_i}}}$$

– *AND gate*
Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{AND}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. Suppose that $f_w(x) = 1$. Then $f_{A(w)}(x) = f_{B(w)}(x) = 1$ and we compute:

$$\begin{aligned} E_w &= e(E_{A(w)}, K_{w,1}) \cdot e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,3}, D(x)) \\ &= e\big((g_{j+n-1})^{r_{A(w)} \prod_i d_{i,x_i}}, g^{a_w}\big) \cdot e\big((g_{j+n-1})^{r_{B(w)} \prod_i d_{i,x_i}}, g^{b_w}\big) \cdot \\ &\qquad \cdot e\big(g_j^{r_w - a_w \cdot r_{A(w)} - c_w \cdot r_{B(w)}}, g_n^{\prod_i d_{i,x_i}}\big) \\ &= (g_{j+n})^{r_w \prod_i d_{i,x_i}} \end{aligned}$$

The procedures above are evaluated in order for all w for which $f_w(x) = 1$. The final output gives $E_{n+q} = F(k, x)$.

5.2 Proof of Security

We now prove security of the circuit constrained construction. We show that for an n-bit domain and circuits of depth ℓ, if the $\kappa = n + \ell$-Multilinear Decisional Diffie-Hellman assumption holds then our construction is secure for appropriate choice of the group generator security parameter.

Our proof begins as in the bit-fixing proof where a where we use the standard complexity leveraging technique of guessing the challenge x^* ahead of time to prove adaptive security. The guess will cause a loss of $1/2^n$ factor in the reduction. The delegate oracle queries, however, are handled quite differently.

Theorem 4. *If there exists a poly-time attack algorithm $\mathcal{A}$ that breaks our circuit constrained construction n-bit input and circuits of depth ℓ with advantage $\epsilon(\lambda)$ there there exists a poly-time algorithm $\mathcal{B}$ that breaks the $\kappa = n + \ell$-Multilinear Decisional Diffie-Hellman assumption with advantage $\epsilon(\lambda)/2^n$.*

Due to space constraints the proof appears in the full version of the paper [8].

6 Applications

Having constructed constrained PRFs for several predicate families we now explore a number of remarkable applications for these concepts. Our primary goal is to demonstrate the versatility and general utility of constrained PRFs.

6.1 Broadcast Encryption with Optimal Ciphertext Length

We start by showing that a bit-fixing constrained PRF leads a broadcast encryption system with *optimal* ciphertext size. Recall that a broadcast encryption system [15] is made up of three randomized algorithms:

Setup(λ, n)**.** Takes as input the security parameter λ and the number of receivers n. It outputs n private keys $d_1, \ldots, d_n$ and a broadcaster key bk. For $i = 1, \ldots, n$, recipient number i is given the private key d_i.

Encrypt(bk, S)**.** Takes as input a subset $S \subseteq \{1, \ldots, n\}$, and the broadcaster's key bk. It outputs a pair (hdr, k) where hdr is called the header and $k \in \mathcal{K}$ is a message encryption key chosen from the key space $\mathcal{K}$. We will often refer to hdr as the broadcast ciphertext.

Let m be a message to be broadcast that should be decipherable precisely by the receivers in S. Let c_m be the encryption of m under the symmetric key k. The broadcast data consists of (S, hdr, c_m). The pair (S, hdr) is often called the full header and c_m is often called the broadcast body.

Decrypt$(i, d_i, S, \mathsf{hdr})$**.** Takes as input a subset $S \subseteq \{1, \ldots, n\}$, a user id $i \in \{1, \ldots, n\}$ and the private key d_i for user i, and a header hdr. If $i \in S$ the algorithm outputs a message encryption key $k \in \mathcal{K}$. Intuitively, user i can then use k to decrypt the broadcast body c_m and obtain the message m.

In what follows the broadcaster's key bk is a secret key known only to the broadcaster and hence our system is a secret-key broadcast encryption.

The **length efficiency** of a broadcast encryption system is measured in the length of the header hdr. The shorter the header the more efficient the system. Remarkably, some systems such as [6,13,12,7,30] achieve a fixed size header that depends only on the security parameter and is independent of the size of the recipient set S.

As usual, we require that the system be correct, namely that for all subsets $S \subseteq \{1, \ldots, n\}$ and all $i \in S$ if $(\mathsf{bk}, (d_1, \ldots, d_n)) \xleftarrow{R} \mathit{Setup}(n)$ and $(\mathsf{hdr}, k) \xleftarrow{R} \mathit{Encrypt}(\mathsf{bk}, S)$ then $\mathit{Decrypt}(i, d_i, S, \mathsf{hdr}) = k$.

A broadcast encryption system is said to be semantically secure if an adaptive adversary $\mathcal{A}$ that obtains recipient keys d_i for $i \in S$ of its choice, cannot break the semantic security of a broadcast ciphertext intended for a recipient set S^* in the complement of S, namely $S^* \subseteq [n] \setminus S$. More precisely, security is defined using the following experiment, denoted $\mathrm{EXP}(b)$, parameterized by the total number of recipients n and by a bit $b \in \{0, 1\}$:

$(\mathsf{bk}, (d_1, \ldots, d_n)) \xleftarrow{R} \mathit{Setup}(\lambda, n)$

$b' \leftarrow \mathcal{A}^{\mathsf{RK}(\cdot), \mathsf{SK}(\cdot), \mathsf{RoR}(b, \cdot)}(\lambda, n)$

where

$\mathsf{RK}(i)$ is a recipient key oracle that takes as input $i \in [n]$ and returns d_i,

$\mathsf{SK}(S)$ takes as input $S \subseteq [n]$ and returns $\mathit{Encrypt}(\mathsf{bk}, S)$, and

$\mathsf{RoR}(b, S^*)$ is a real-or-random oracle: it takes as input $b \in \{0, 1\}$ and $S^* \subseteq [n]$, computes $(\mathsf{hdr}, k_0) \xleftarrow{R} \mathit{Encrypt}(\mathsf{bk}, S^*)$ and $k_1 \xleftarrow{R} \mathcal{K}$, and returns (hdr, k_b).

We require that all sets S^* given as input to oracle RoR are distinct from all sets S given as input to SK and that S^* does not contain any index i given as input to RK. For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\mathrm{EXP}(b)$ and as usual define $\mathrm{AdvBE}_{\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 2. *We say that a broadcast encryption is semantically secure if for all probabilistic polynomial time adversaries $\mathcal{A}$ the function $AdvBE_{\mathcal{A}}(\lambda)$ is negligible.*

An length-optimal broadcast encryption construction. A bit-fixing PRF such as the one constructed in Section 4 gives a broadcast encryption system with optimal ciphertext length. Specifically, the header size is always 0 for all recipient sets $S \subseteq [n]$. The system, denoted BE_F works as follows:

Setup(λ, n)**:** Let $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ be a secure bit-fixing constrained PRF. Choose a random key $\mathsf{bk} \xleftarrow{R} \mathcal{K}$ and for $i = 1, \ldots, n$ compute

$$d_i \leftarrow F.\mathsf{constrain}(\mathsf{bk},\ p_i)$$

where $p_i : \{0,1\}^n \to \{0,1\}$ is the bit-fixing predicate satisfying $p_i(x) = 1$ iff $x_i = 1$. Thus, the key d_i enables the evaluation of $F(\mathsf{bk}, x)$ at any point $x \in \{0,1\}^n$ for which $x_i = 1$. Output $(\mathsf{bk}, (d_1, \ldots, d_n))$.

Encrypt(bk, S)**:** Let $x \in \{0,1\}^n$ be the characteristic vector of S and compute $k \leftarrow F(\mathsf{bk}, x)$. Output the pair (hdr, k) where $\mathsf{hdr} = \epsilon$. That is, the output header is simply the empty string.

Decrypt$(i, d_i, S, \mathsf{hdr})$**:** Let $x \in \{0,1\}^n$ be the characteristic vector of S. If $i \in S$ then the bit-fixing predicate p_i satisfies $p_i(x) = 1$. Therefore, d_i can be used to compute $F(\mathsf{bk}, x)$, as required.

Theorem 5. *BE_F is a semantically secure broadcast encryption system against adaptive adversaries assuming that the underlying constrained bit-fixing PRF is secure.*

Proof. Security follows immediately from the security of the bit-fixing PRF. Specifically, oracle RK in the broadcast encryption experiment is implemented using oracle $F.\mathsf{constrain}$ in the constrained security game (Section 3.2). Oracle SK is implemented using oracle $F.\mathsf{eval}$ in the constrained security game. Finally, the broadcast encryption real-or-random oracle is the same as the $\mathsf{Challenge}$ oracle in the constrained security game. Therefore, an attacker who succeeds in breaking semantic security of the broadcast encryption system will break security of the bit-fixing PRF.

Comparison to existing fully collusion resistant schemes. While our primary goal is to illustrate applications of abstract constrained PRFs, it is instructive to examine the specific broadcast system that results from instantiating the system above with the bit-fixing PRF in Section 4. We briefly compare this system to existing broadcast encryption systems such as [6,13,12,30]. These existing systems are built from bilinear maps, they allow the broadcaster's key to be public, and the broadcast header contains a constant number of group elements. The benefit of the instantiated system above is that the header length is smaller: its length is zero. However, the system uses multi-linear maps and the broadcaster's key is secret. The system is closely related to the multilinear-based broadcast system

of Boneh and Silverberg [7] which has similar parameters. To re-iterate, our goal is to show the general utility of constrained PRFs. Nevertheless, we hope that future constrained PRFs will lead to new families of broadcast systems.

6.2 Identity-Based Key Exchange

Next, we show that a left/right constrained PRF directly implies an identity-based non-interactive key exchange (ID-NIKE) system [31,14,27,16]. Recall that such a system is made up of three algorithms:

- *Setup*(λ) outputs public parameters pp and a master secret msk,
- *Extract*($\mathsf{msk}, \mathrm{id}$) generates a secret key $\mathsf{sk}_{\mathrm{id}}$ for identity id, and
- *KeyGen*($\mathsf{pp}, \mathsf{sk}_{\mathrm{id}}, \mathrm{id}'$) outputs a shared key $k_{\mathrm{id},\mathrm{id}'}$.

For correctness we require that $\mathit{KeyGen}(\mathsf{pp}, \mathsf{sk}_{\mathrm{id}}, \mathrm{id}') = \mathit{KeyGen}(\mathsf{pp}, \mathsf{sk}_{\mathrm{id}'}, \mathrm{id})$ for all $\mathrm{id} \neq \mathrm{id}'$ and pp generated by *Setup*.

Briefly, the security requirement, defined by Dupont and Enge [14] and further refined by Paterson and Srinivasan [27], is that an adversary $\mathcal{A}$ who obtains secret keys $\mathsf{sk}_{\mathrm{id}}$ for all identities $\mathrm{id} \in S$ for a set S of his choice, cannot distinguish the shared key $k_{\mathrm{id}_*,\mathrm{id}'_*}$ from random for identities $\mathrm{id}_*, \mathrm{id}'_* \notin S$ of his choice. The adversary may also ask to reveal the shared key $k_{\mathrm{id},\mathrm{id}'}$ for any pair of identities $(\mathrm{id}, \mathrm{id}') \neq (\mathrm{id}_*, \mathrm{id}'_*)$.

Identity-based key exchange from left/right constrained PRFs. The system works as follows:

- *Setup*(λ) : let $F : \mathcal{K} \times \mathcal{X}^2 \to \mathcal{Y}$ be a secure left/right constrained PRF. Choose a random $\mathsf{msk} \xleftarrow{R} \mathcal{K}$ and output msk. The public parameters pp are the (optional) public parameters of the PRF.
- *Extract*($\mathsf{msk}, \mathrm{id}$): compute $d_{\mathrm{L}} = F.\mathsf{constrain}(\mathsf{msk}, p^{(\mathrm{L})}_{\mathrm{id}})$ and $d_{\mathrm{R}} = F.\mathsf{constrain}(\mathsf{msk}, p^{(\mathrm{R})}_{\mathrm{id}})$. Output $\mathsf{sk}_{\mathrm{id}} = (d_{\mathrm{L}}, d_{\mathrm{R}})$.
- *KeyGen*($\mathsf{sk}_{\mathrm{id}}, \mathrm{id}'$): We assume that the identity strings are lexicographically ordered. Output $k_{\mathrm{id},\mathrm{id}'} = F\big(\mathsf{msk}, (\mathrm{id}, \mathrm{id}')\big)$ if $\mathrm{id} < \mathrm{id}'$ using d_{L}. Output $k_{\mathrm{id},\mathrm{id}'} = F\big(\mathsf{msk}, (\mathrm{id}', \mathrm{id})\big)$ if $\mathrm{id} > \mathrm{id}'$ using d_{R}. By definition of a left/right constrained PRF, both values can be computed just given $\mathsf{sk}_{\mathrm{id}}$.

Correctness of the system follows directly from the correctness of the constrained PRF and lexicographic convention. Security again follows directly from the security definition of a constrained PRF. Oracle $F.\mathsf{constrain}$ in the constrained security game (Section 3.2) enables the adversary $\mathcal{A}$ to request the secret keys for any set of identities S of her choice. Oracle $F.\mathsf{eval}$ enables the adversary $\mathcal{A}$ to reveal the shared key $k_{\mathrm{id},\mathrm{id}'}$ for any pair of identities $(\mathrm{id}, \mathrm{id}')$. If $\mathcal{A}$ could then distinguish $F(\mathsf{msk}, (\mathrm{id}_*, \mathrm{id}'_*))$ from random for some $\mathrm{id}_*, \mathrm{id}'_* \notin S$ and for which reveal was not called then she would solve the challenge in the constrained security game.

Comparison to existing ID-NIKE. While our primary goal is to explore applications of general constrained PRFs, it is instructive to examine the specific ID-NIKE systems obtained by instantiating the ID-NIKE above with our specific PRFs. The first concrete ID-NIKE is obtained from the left/right constrained PRF in Eq. (1). This ID-NIKE is identical to the Sakai-Ohgishi-Kasahara [31] ID-NIKE which was analyzed in [14,27]. A second ID-NIKE is obtained by using the bit-fixing constrained PRF in Section 4 as a left/right constrained PRF. The resulting ID-NIKE is related to a recent ID-NIKE due to Freire et al. [16] which is the first ID-NIKE proven secure in the standard model. While *KeyGen* in our instantiated ID-NIKE uses fewer group elements than [16], we achieve adaptive security via complexity leveraging which forces our multilinear groups to be substantially larger. This likely results in an overall less efficient ID-NIKE when compared to [16].

As stated above, our primary goal here is to explore the power of constrained PRFs. We hope that future constrained PRFs, especially ones built from the learning with errors (LWE) assumption, will give new ID-NIKE systems.

6.3 Policy-Based Key Distribution

More generally, our constrained PRF construction for circuit predicates (Section 5) gives rise to a powerful non-interactive group key distribution mechanism.

Suppose each user in the system is identified by a vector $\mathrm{id} \in \{0,1\}^n$ that encodes a set of attributes for that user. Our goal is that for any predicate $p : \{0,1\}^n \to \{0,1\}$, users whose id satisfies $p(\mathrm{id}) = 1$ will be able to compute a shared key k_p. However, a coalition of users for which $p(\mathrm{id}) = 0$ for all members of the coalition learns nothing about k_p. We call this mechanism *non-interactive policy-based key exchange* (PB-NIKE) since only those users whose set of attributes satisfies the policy p are able to compute the shared key k_p.

For example, consider the policy p that is true for users who are members of the IACR and have a driver's license. All such users will be able to derive the policy shared key k_p, but to all other users the key k_p will be indistinguishable from random. This k_p can then be used for secure communication among the group members. This functionality is related to the concept of Attribute-Based Encryption [29,22].

We implement policy-based key agreement using a constrained PRF $F : \mathcal{K} \times \{0,1\}^m \to \mathcal{Y}$ for circuit predicates. To do so, let $U(\cdot,\cdot)$ denote a universal circuit that takes two inputs: an identity $\mathrm{id} \in \{0,1\}^n$ and an m-bit description of a circuit for a predicate $p : \{0,1\}^n \to \{0,1\}$. The universal circuit $U(\mathrm{id}, p)$ is defined as:

$$U(\mathrm{id}, p) = p(\mathrm{id}) \in \{0,1\}$$

We define the secret key $\mathsf{sk}_{\mathrm{id}}$ given to user id to be the constrained PRF key that lets user id evaluate $F(\mathsf{msk}, p)$ for all p for which $U(\mathrm{id}, p) = p(\mathrm{id}) = 1$. Thus, users whose set of attributes id satisfies $p(\mathrm{id}) = 1$ can compute the policy key $k_p = F(\mathsf{msk}, p)$ using their secret key $\mathsf{sk}_{\mathrm{id}}$. All other users cannot.

In more detail, the system works as follows:

- *Setup*(λ) : let $F : \mathcal{K} \times \{0,1\}^m \to \mathcal{Y}$ be a secure constrained PRF for circuit predicates. The master secret msk is chosen as a random key in $\mathcal{K}$.
- *Extract*($\mathsf{msk}, \mathrm{id}$): output $\mathsf{sk}_{\mathrm{id}} = F.\mathsf{constrain}\big(\mathsf{msk},\ U(\mathrm{id}, \cdot)\ \big)$. By definition, this key $\mathsf{sk}_{\mathrm{id}}$ enables the evaluation of $F(\mathsf{msk}, p)$ at all p such that $U(\mathrm{id}, p) = p(\mathrm{id}) = 1$, as required.

The properties of F imply that for any predicate p (whose description is at most m bits), the group key $k_p = F(\mathsf{msk}, p)$ can be computed by any user whose id satisfies $p(\mathrm{id}) = 1$. Moreover, the security property for constrained PRFs implies that a coalition of users for which $p(\mathrm{id}) = 0$ for all members of the coalition cannot distinguish k_p from random.

7 Extensions and Open Problems

We constructed constrained PRFs for several natural predicate families and showed applications for all these constructions. Here we point out a few possible directions for future research.

First, it would be interesting to generalize the constrained concept to allow for multiple levels of delegation. That is, the master key for the PRF can be used to derive a constrained key k_{s} for some set $S \subset \mathcal{X}$. That key k_{s} can be used in turn to derive a further constrained key k'_{s} for some subset $S' \subset S$, and so on. This concept is in similar spirit to Hierarchical IBE [23,19,10] or delegation in ABE [22]. For the GGM prefix system, this is straightforward. Some of our constructions, such as the bit fixing PRF, extend naturally to support more than one level of delegation while others do not.

Second, for the most interesting predicate families our constructions are based on multilinear maps. It would be quite useful to provide constructions based on other assumptions such as Learning With Errors (LWE) or simple bilinear maps.

Acknowledgments. Dan Boneh is supported by NSF, the DARPA PROCEED program, an AFO SR MURI award, a grant from ONR, an IARPA project provided via DoI/NBC, and by a Google faculty research award.

Brent Waters is supported by NSF CNS-0915361, CNS-0952692, CNS-1228599, DARPA N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or IARPA.

References

1. Bellare, M., Fuchsbauer, G.: Policy-based signatures. Cryptology ePrint Archive, Report 2013/413 (2013)

2. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM Conference on Computer and Communications Security, pp. 784–796 (2012)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. 32(3), 586–615 (2001); Extended abstract in Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
7. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics 324, 71–90 (2003)
8. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2013/352 (2013)
9. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401 (2013)
10. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
11. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. Cryptology ePrint Archive, Report 2013/183 (2013)
12. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)
13. Delerablée, C., Paillier, P., Pointcheval, D.: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–59. Springer, Heidelberg (2007)
14. Dupont, R., Enge, A.: Provably secure non-interactive key distribution based on pairings. Discrete Applied Mathematics 154(2), 270–276 (2006)
15. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
16. Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 513–530. Springer, Heidelberg (2013)
17. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
18. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013)
19. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
20. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 34(4), 792–807 (1986)

21. Choudary Gorantla, M., Boyd, C., Nieto, J.M.G.: Attribute-based authenticated key exchange. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 300–317. Springer, Heidelberg (2010)
22. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
23. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
24. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Proceedings ACM CCS (2013)
25. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
26. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: FOCS 1997, pp. 458–467 (1997)
27. Paterson, K., Srinivasan, S.: On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. Des. Codes Cryptography 52(2), 219–241 (2009)
28. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proc. of STOC 2005, pp. 84–93 (2005)
29. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Sakai, R., Furukawa, J.: Identity-based broadcast encryption. Cryptology ePrint Archive, Report 2007/217 (2007)
31. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS (2000)

A Circuit Notation

We now define our notation for circuits that adapts the model and notation of Bellare, Hoang, and Rogaway [2] (Section 2.3). For our application we restrict our consideration to certain classes of boolean circuits. First, our circuits will have a single output gate. Next, we will consider layered circuits. In a layered circuit a gate at depth j will receive both of its inputs from wires at depth $j-1$. Finally, we will restrict ourselves to monotonic circuits where gates are either AND or OR gates of two inputs. [3]

Our circuits will be a five tuple $f = (n, q, A, B, \texttt{GateType})$. We let n be the number of inputs and q be the number of gates. We define inputs $= \{1, \ldots, n\}$, Wires $= \{1, \ldots, n+q\}$, and Gates $= \{n+1, \ldots, n+q\}$. The wire $n+q$ is the designated output wire. A : Gates $\rightarrow$ Wires/outputwire is a function where $A(w)$ identifies w's first incoming wire and B : Gates $\rightarrow$ Wires/outputwire is a function where $B(w)$ identifies w's second incoming wire. Finally, $\texttt{GateType}$:

[3] These restrictions are mostly useful for exposition and do not impact functionality. General circuits can be built from non-monotonic circuits. In addition, given a circuit an equivalent layered exists that is larger by at most a polynomial factor.

Gates $\rightarrow \{\text{AND}, \text{OR}\}$ is a function that identifies a gate as either an AND or OR gate.

We require that $w > B(w) > A(w)$. We also define a function $\texttt{depth}(w)$ where if $w \in$ inputs $\texttt{depth}(w) = 1$ and in general $\texttt{depth}(w)$ of wire w is equal to the shortest path to an input wire plus 1. Since our circuit is layered we require that for all $w \in$ Gates that if $\texttt{depth}(w) = j$ then $\texttt{depth}(A(w)) = \texttt{depth}(B(w)) = j-1$.

We will abuse notation and let $f(x)$ be the evaluation of the circuit f on input $x \in \{0,1\}^n$. In addition, we let $f_w(x)$ be the value of wire w of the circuit on input x.